Este exemplar corresponde à redação final da Tese/Dissertação devidamente corrigida e defendida por: Sandra Regina Rocha

e aprovada pela Banca Examinadora.

Campinas, 02 de Junho de 2003

COORDENADOR DE PÓS-GRADUAÇÃO
CPG-IC

Análise de Desempenho de Junções Espaciais: Uma Comparação entre os Métodos Analítico e Experimental

Sandra Regina Rocha

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Análise de Desempenho de Junções Espaciais: Uma Comparação entre os Métodos Analítico e Experimental

Sandra Regina Rocha

Fevereiro de 2003

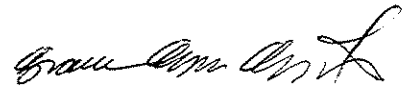
Banca Examinadora:

- Geovane Cayres Magalhães
  Instituto de Computação, Unicamp (Ph.D.)
- Ricardo Rodrigues Ciferri, Doutor
  Departamento de Informática, Universidade Estadual de Maringá
- Célio Cardoso Guimarães
  Instituto de Computação, Unicamp
- Neucimar Geronimo Leite, Ph.D.
  Instituto de Computação, Unicamp

Análise de Desempenho de Junções Espaciais: Uma Comparação entre os Métodos Analítico e Experimental

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Sandra Regina Rocha e aprovada pela Banca Examinadora.

Campinas, 27 de Fevereiro de 2003.



Geovane Cayres Magalhães
Instituto de Computação, Unicamp (Ph.D.)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Rocha, Sandra Regina

R582a Análise de desempenho de junções espaciais: uma comparação entre os métodos analítico e experimental / Sandra Regina Rocha -- Campinas, [S.P. :s.n.], 2003.

Orientadores : Geovane Cayres Magalhães.

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Sistemas de informação geográfica. 2. Desempenho. 3. Banco de dados. I. Magalhães, Geovane Cayres. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 27 de fevereiro de 2003, pela Banca examinadora composta pelos Professores Doutores:

Ricardo Rodrigues Ciferri

Prof. Dr. Ricardo Rodrigues Ciferri
UEM

Prof. Dr. Célio Cardoso Guimarães
IC - UNICAMP

Prof. Dr. Geovane Cayres Magalhães
IC – UNICAMP

Dedicatória

A minha querida amiga Regina Sugai, que apesar de estar em Roma, está sempre presente na minha vida pela comunhão dos Santos. Ao meu diretor espiritual Pe. Ricardo por ter sempre a santa paciência de me escutar e por acreditar em mim. A minha irmã Cíntia Viviane Rocha, a meus pais, a meus irmãos e a meus padrinhos por tudo que são...

Esta tese é dedicada a todos meus familiares e amigos. Especialmente aos amigos do Opus Dei e do PUR/GOU(RCC). E a todos os corações jovens do mundo que tem a coragem, a alegria e o orgulho de vestir a camisa da geração João PauloII.

Aos amigos Carlos Eduardo Fredo e Nelson Uto por serem o diferente na minha vida.

Nem todos podem chegar a ser ricos, sábios, famosos... Em contrapartida, todos - sim, "todos" - estão chamados a serem santos. (Ponto 125 - Sulco)

É necessário estudar... Mas não é suficiente. Que se pode conseguir de quem se esfalfa para alimentar o seu egoísmo, ou de quem não persegue outro objetivo senão o de garantir a tranquilidade, para daqui a alguns anos?

É preciso estudar..., para ganhar o mundo e conquistá-lo para Deus. Então elevaremos o nível do nosso esforço, procurando que o trabalho realizado se converta em encontro com o Senhor, e sirva de basa a outros, aos que seguirão o nosso caminho...

- Deste modo o estudo será oração.

(Ponto 526 - Sulco)

Não percas nunca de vista a mira sobrenatural. - Retifica a intenção, como se vai retificando o rumo do navio no mar alto: olhando para o estrela, olhando para Maria. E terás a certeza de chegar sempre a bom porto.

(Ponto 749 - Forja)

São JoséMaria Escrivá
fundador do Opus Dei

Agradecimentos

Agradeço primeiramente a Deus, a Nossa Senhora e a meu Anjo da Guarda.

Agradecimentos especiais ao professor Geovane Cayres Magalhães pela orientação, a professora Claudia Bauzer por organizar o grupo de banco de dados e ao professor Ricardo Rodrigues Ciferri pela co-orientação informal e pelo apoio técnico.

Agradeço ao Nelson Uto, Handall Luis Adam, Carlos Eduardo Fredo, Daniel da Silva Andrade, Vera Regazzi, Ana Márcia Taveira e Sandro Gatti pelas dicas e apoio dados durante a pesquisa.

Agradeço ao laboratório de banco de dados da Unicamp (LIS) pelo fornecimento da infraestrutura, ao Centro de Pesquisa e Desenvolvimento (CPqD) pela liberação do conjunto de dados reais para nossos testes e a Universidade Estadual de Maringá pelo apoio técnico.

Resumo

Este trabalho aplica e avalia os métodos analíticos para desempenho de junções espaciais baseadas em indexação por árvores-R e compara os resultados obtidos com os resultados provenientes de uma implementação que utiliza dados sintéticos e reais.

Nosso foco é a análise de desempenho da fase de filtragem de junções espaciais de interseção sobre dados espaciais indexados por estruturas derivadas de árvore-R* CR . O estudo é feito para aplicações geográficas em espaços bidimensionais, tendo seus objetos espaciais aproximados por seus retângulos envolventes mínimos para a fase de filtragem da junção espacial. É feita uma investigação comparativa do desempenho calculado por um método analítico atual e os resultados provenientes de uma implementação aplicada a dados sintéticos e dados reais.

O objetivo deste trabalho é avaliar os métodos analíticos presentes na literatura. A avaliação da qualidade do método é medida baseando-se na comparação entre os resultados analíticos e os resultados experimentais. Os dados sintéticos de entrada para os testes seguem um modelo de distribuição de dados espaciais inicialmente definido por Ciferri [6] e também foram gerados conjuntos de dados sintéticos compostos de quadrados com dimensão constante equivalente a 2% do *extend* simulando distribuição uniorme no *extent*, além de um conjunto de dados reais. Pretende-se verificar qual a influência do fator distribuição dos dados na qualidade do valor obtido pelo método analítico mais conhecido atualmente proposto na literatura.

Este trabalho é uma extensão ao trabalho de Gatti [12]. Os resultados experimentais foram obtidos através de uma implementação baseada no modelo de implementação para junções espaciais definido por Gatti [12]. No trabalho de Gatti[12] foi feita uma análise de desempenho de junções espaciais baseada nos fatores que afetam o desempenho da junção, tais como o tamanho do *buffer*, o tamanho das páginas de disco entre outros. O trabalho de Gatti [12] não faz uma projeção de resultados baseados em modelos analíticos, além disso o trabalho de Gatti [12] utilizou apenas dados reais para sua análise. O presente trabalho utiliza também dados sintéticos.

Para o cálculo do número de acessos a disco para as junções espaciais, o fator distribuição espacial dos dados tem um grande impacto na qualidade do valor obtido pelo

método analítico mais atual proposto na literatura (método de Theodoridis [36]). Com base nos resultados obtidos, concluimos que o método analítico de Theodoridis [36] apresenta uma boa qualidade de resultado (pequena diferença entre valor analítico e experimental) para distribuições uniformes no espaço de objetos espaciais com forma quadrada. Porém ele deve ser adaptado para poder ser utilizado para um distribuição espacial genérica. Essa grande dependência do fator distribuição espacial para o método analítico de Theodoridis [36] pode ser justificada pelo fato do método analítico de Theodoridis [36] ter sido construído assumindo-se algumas premisssas que estão vinculadas a distribuição espacial dos dados. Por exemplo, o modelo analítico de Theodoridis [36] assume como premissa que os dados espaciais são quadrados e que a densidade global do espaço reflete a denside do espaço em cada sub-área do mesmo. Com base nos resultados obtidos pelos trabalhos de Theodoridis [36] e Huang [15], foi considerada que a diferença entre o valor analítico e experimental era pequena quando a divisão entre o resultado analítico sobre o valor experimental fosse menor que 0,3.

Abstract

This work analyses the quality of Theodoridis cost model [36] for join queries. Theodoridis [36] cost model is the more recent and accepted cost model for predict the performace based on dick access number in the filtering phase of intersection spatial joins queries present in the literature. We compare Theodoridis [36] cost model results with the results provided by an implementation using real and synthetic data to estimate the quality of the cost model. This implementation uses the R-tree* CR index structure. We use synthetic data with different space distribution and real data. Our goal is to investigate the influence of the data distribution factor in the quality of the Theodoridis [36] cost model. We conclude that the cost model results are strongly influenced by the data space distribution factor.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Junções são operações fundamentais para as consultas nos sistemas de gerenciamento de banco de dados. Banco de dados convencionais utilizam métodos de indexação de dados para melhorar o desempenho das junções. Os métodos tradicionais de indexação não apresentam desempenho satisfatório quando aplicados a dados espaciais de dimensão maior que um [9]. As aplicações espaciais utilizam, além dos dados convencionais, dados espaciais. Os dados espaciais têm uma complexidade de manipulação maior que dados convencionais pois os dados espaciais não possuem uma ordenação natural total que preserve a proximidade espacial e os predicados analisados nas junções espaciais são mais complexos que os analisados nas junções convencionais.

Com a crescente utilização de aplicações geográficas que manipulam dados espaciais torna-se necessário à aplicação de métodos não convencionais para melhorar o desempenho destas aplicações. As estruturas baseadas em árvores-R são apropriadas para indexar dados espaciais [14] pois as árvores-R têm um grande *fan-out (branching factor)*, possibilitando definir o *fan-out* equivalente a capacidade de armazenamento de uma página de disco, as árvores-R são balanceadas, e os dados são armazenados mantendo sua relação de proximidade espacial . O processo de junção baseado em árvores-R apresenta resultados superiores de desempenho quando comparado a outras estruturas de indexação [32].

Sistemas de informação geográficos como *Intergraph´s , DBSs, Postgres, MapInfo, Informix [1] e Oracle Spatial [18]* usam árvores-R como seu método básico de acesso a dados gerando um grande interesse em estudar as junções espaciais baseadas em árvores da família R.

Devido a grande complexidade de se verificar diretamente um predicado espacial sobre a representação geométrica exata de um objeto espacial utiliza-se aproximações simplificadoras para os objetos espaciais nas aplicações geográficas. Uma solução muito comum é a utilização de uma aproximação conservativa onde cada ponto do contorno do objeto original está também na aproximação conservativa. Uma aproximação conservativa muito

utilizada pelas aplicações geográficas é o retângulo envolvente mínimo *(minimum boundig retangle MBR)* que será a aproximação utilizada no presente estudo.

O presente estudo enfocará as junções espaciais onde todas as relações da junção estão indexadas por um método de indexação baseado em árvores-R. As junções serão realizadas tomando as relações duas a duas *(2-way R-tree join)* [4]. Dado um conjunto de relações espaciais indexadas por um método derivado das árvores da família R é possível melhorar o desempenho da junção desses objetos espaciais através de técnicas aplicadas aos algoritmos que percorrem sincronamente as árvores R das relações. Para diminuir o tempo de *CPU* empregam-se as técnicas de restrição do espaço de busca *(search space restriction)* e varredura de planos *(plane sweeping)* e para diminuir o número de acessos a disco *(IO-time)* emprega-se a política de fixação das páginas de *buffer (buffer pinning)*.

Na busca em largura descrita na seção 2.11 todos os nós de um mesmo nível da árvore podem ser consultados em uma mesma interação possibilitando a construção de índices intermediários para a junção. O gerenciamento dos índices intermediários de junção possibilita uma melhora na previsão das páginas que devem ser fixadas no *buffer*. A busca em largura permite a aplicação da otimização global baseada nos índices intermediários, tais como a ordenação dos índices intermediários, o gerenciamento de memória e o gerenciamento de *buffer*.

A fase de testes dos métodos de junção espacial utilizará dados provenientes de uma aplicação real e dados sintéticos. Como uma extensão proposta ao trabalho de Gatti [12], os resultados obtidos destes testes serão comparados com os resultados obtidos por métodos analíticos propostos na literatura [35] [34]. O objetivo deste estudo comparativo é de validar a aplicabilidade dos métodos analíticos para junções no caso específico de junções com dados espaciais representados em espaços de poucas dimensões. Além disso pretende-se comparar o comportamento desses métodos analíticos quando aplicados a implementações com dados reais e sintéticos. O método analítico utilizado será o proposto em Theodoridis [36] que é um método que depende apenas do número de objetos espaciais e da densidade do espaço de dados. Este método analítico foi desenvolvido para avaliar a fase de filtragem de uma junção espacial que utiliza predicados espaciais topológicos de interseção onde os objetos espaciais estão aproximados por seus *MBR* e são indexados por uma estrutura baseada nas árvores da família R.

1.1 Motivação

A crescente utilização de aplicações que utilizam dados espaciais (tais como pontos, linhas, retângulos e polígonos) tem despertado um grande interesse no estudo do desempenho de sistemas gerenciadores de banco de dados (SGBDs) espaciais e especialmente no estudo do desempenho de estruturas de indexação espacial. O grande número de estruturas de

indexação espacial propostas atualmente na literatura indica uma necessidade intrínseca de se desenvolver e de se estudar modelos analíticos que predigam o desempenho destas estruturas de indexação. Em particular, a motivação para o estudo de modelos analíticos reside no fato destes modelos permitirem compreender melhor o comportamento de uma estrutura de indexação quando aplicada a conjuntos de dados espaciais distintos com diferentes volumes e características de tamanho e formato. Adicionalmente, os modelos analíticos podem ser usados objetivamente como um ponto de comparação entre várias propostas de estruturas de indexação espacial. Além do mais, modelos analíticos são freqüentemente utilizados pelo otimizador de consultas de um SGBD espacial para avaliar o custo de uma consulta espacial complexa, com o propósito de definir uma execução mais eficiente para a consulta [37].

Esta pesquisa tem por objetivo principal comparar os resultados de desempenho propiciados por modelos analíticos e por modelos experimentais. No contexto deste trabalho, ambos os modelos referem-se à investigação do desempenho de técnicas de junção espacial baseadas em árvores-R (uma família particular de estruturas de indexação espacial).

Mais especificamente, esta pesquisa tem por objetivo avaliar e aplicar modelos analíticos, assim como avaliar e aplicar modelos experimentais propostos na literatura. Posteriormente, objetiva-se comparar os resultados de desempenho produzidos pelos modelos analíticos com os resultados de desempenho obtidos em testes experimentais. A investigação será direcionada para se determinar à correlação existente entre a distribuição espacial dos dados e a qualidade dos resultados de desempenho produzidos pelos modelos analíticos e experimentais. Assim, por exemplo, será possível definir os casos nos quais os modelos analíticos podem ser aplicados com maior precisão na previsão do desempenho de técnicas de junção espacial baseadas em árvores-R. A comparação entre os resultados de desempenho produzidos por modelos analíticos e experimentais permitirá verificar se existe a necessidade de alteração dos modelos analíticos estudados ou a necessidade de se definir um novo modelo analítico mais preciso.

Com relação aos modelos analíticos, nós estamos investigando o modelo proposto por Theodoridis [36]. Em particular, nós almejamos validar e aplicar este modelo no caso específico de junção espacial de interseção com dados espaciais representados em espaços de baixa dimensionalidade (tipicamente, o espaço Euclidiano bidimensional). Segundo o nosso conhecimento, não existe na literatura nenhum trabalho que compare os modelos analíticos voltados para analisar o desempenho de junções espaciais. Também desconhecemos a existência de qualquer estudo que mostre o efeito que o fator distribuição espacial dos dados exerce sobre os resultados de desempenho calculados por modelos analíticos. A nossa pesquisa, neste sentido, mostra-se original e pretende realizar estes estudos inexistentes.

Com relação à abordagem experimental, nós estamos estudando o modelo definido em

Gatti [12]. No trabalho de Gatti, foi feita uma análise de desempenho experimental de técnicas de junção espacial baseadas em árvores-R. Os dados utilizados no referido trabalho, no entanto, limitaram-se a conjuntos de dados reais relativos à cidade de Valinhos-SP no contexto do projeto SAGRE [6]. A nossa pesquisa pretende estender o estudo realizado por Gatti utilizando dados sintéticos nos testes de desempenho experimentais. Almeja-se, desta forma, realizar uma investigação experimental mais abrangente do desempenho de técnicas de junção espacial baseadas em árvores-R. Em especial, a finalidade de se utilizar dados sintéticos é que estes dados permitem o controle do fator distribuição espacial dos dados, permitindo assim que se faça uma investigação do impacto deste fator no desempenho das técnicas de junção espacial.

Citamos também que os modelos experimentais têm sido amplamente utilizados na análise de desempenho de estruturas de indexação espacial, com ênfase no uso da técnica experimental de benchmark de banco de dados [2, 9, 7, 5, 26, 8, 12, 6, 13]. Esta técnica consiste da execução de um conjunto conhecido de consultas espaciais e de operações de inserção, remoção e modificação de dados (i.e., a sua carga de trabalho) em um conjunto de dados espaciais também conhecido e em geral gerado artificialmente. Vale destacar que os resultados propiciados pela técnica de benchmark são altamente confiáveis, uma vez que o próprio sistema sendo analisado (i.e., a estrutura de indexação espacial) é usado para a obtenção dos resultados de desempenho. Além do mais, modelos experimentais capturam custos freqüentemente ignorados ou subestimados em modelos analíticos.

1.2 Organização da Tese

Neste capítulo apresentamos a introdução e a motivação para esta tese. O capitulo 2 apresenta a introdução teórica para as junções espaciais. O capitulo 3 apresenta um *survey* dos trabalhos presentes na literatura relacionados com os métodos analíticos para previsão do desempenho de junções espaciais baseadas em indexação por árvores da família *R*. O capítulo 4 descreve o modelo de implementação utilizado para os testes. No capitulo 5 apresentamos os resultados e a análise comparativa dos mesmos. No capitulo 6 temos a conclusão e apresentamos as extensões para o presente trabalho.

Capítulo 2

Conceitos Básicos

2.1 Objetos espaciais

Objetos espaciais são pontos, linhas, retângulos, polígonos, superfícies e objetos até mais complexos compostos dos mais simples. Os objetos espaciais apresentam uma grande complexidade com relação aos parâmetros número de pontos ou vértices, extensão, forma e sua distribuição no espaço de dados o que implica em um custo alto de avaliação de predicados espaciais sobre esses objetos espaciais. Dentre os dados espaciais, temos muitos aplicações que utilizam os dados geográficos. Os dados geograficos possuem natureza dual, pois um dado geográfico possui uma localização geográfica expressa como coordenadas em um espaço geográfico e atributos descritivos que são representados num banco de dados.

Dados espaciais tem uma complexidade de manipulação maior que dados convencionais, pois os dados espaciais não possuem uma ordenação natural total que preserve a proximidade espacial e os predicados analisados nas junções espaciais são mais complexos que os analisados nas junções convencionais. Por isso os métodos tradicionais de junção amplamente estudados na literatura como o *nested loop*, o *hash join* e o *sort merge* não apresentam um desempenho satisfatório quando aplicados a objetos espaciais. No *nested loop* cada objeto de uma relação tem que ser combinado com todos objetos da outra relação. Para objetos espaciais o desempenho de aplicar o *nested loop* torna-se inadequado já que as relações têm muitos objetos espaciais e o custo de avaliação de um predicado espacial é alto. O algoritmo de junção baseado em *hash* não deve ser aplicado a dados espaciais, já que o *hash* não preserva a ordem espacial, o que implica que com grande probabilidade objetos próximos no espaço não seriam armazenados no mesmo *bucket* degradando o desempenho da junção para dados espaciais.

2.2 Aproximações

Os objetos presentes nas aplicações geográficas podem assumir várias formas. Avaliar um predicado espacial sobre a representação exata de um objeto espacial é um processo muito custoso que envolve a aplicação de algoritmos espaciais. Por isso as aplicações espaciais utilizam aproximações dos objetos que são utilizadas para reduzir o número de objetos que terão sua geometria exata analisada para o predicado. Os objetos espaciais são armazenados na base de dados junto com suas aproximações calculadas. Existem três tipos básicos de aproximações espaciais: as aproximações conservativas, as aproximações progressivas e as aproximações de generalização.

Na fase de filtragem se for utilizada uma estrutura de armazenamento baseada em árvores multiárias para as aproximações e as referências aos objetos espaciais, as aproximações mais complexas, que necessitam de um grande número de parâmetros de descrição, apresentam a desvantagem de ocuparem muito espaço, diminuindo portanto a quantidade de entradas que cabe em cada nó, implicando em um aumento da altura da árvore o que degrada o desempenho das consultas espaciais.

2.2.1 Aproximações conservativas

Uma aproximação é dita conservativa se, e somente se, a fronteira do objeto original está inteiramente contida na aproximação. Se duas aproximações conservativas de objetos espaciais não se interceptam então isto implica que estes objetos não possuem interseção. Uma aproximação conservativa pode ser utilizada no lugar do objeto espacial que ela representa com a finalidade de identificar objetos que não atendem ao predicado de sobreposição e para determinar um outro conjunto de objetos candidatos a atender ao predicado de sobreposição. São exemplos de aproximações conservativas:

- *Minimum bounding rectangle (MBR)* - A aproximação *MBR* é a chave geométrica mais popular. A utilização do *MBR* reduz a complexidade do objeto espacial para 4 parâmetros que contém as características mais importantes do objeto que representam sua posição e sua extensão.

- *Rotated minimum bounding rectangle (RMBR)*- A aproximação *RMBR* é um *MBR* rotacionado que é armazenado através de 5 parâmetros. O calculo do *RMBR* para um objeto geográfico é computado por um algoritmo $O(n^2)$, onde n é o número de vértices do objeto.

- *Convex hull (CH)* - A construção do casco convexo é um problema bastante estudado de geometria computacional que pode ser calculado em $O(n \log n)$, onde n

é o número de vértices do objeto. A quantidade de parâmetros necessária para armazenar o casco convexo é determinada pela geometria do objeto espacial.

- *Minimum bounding m-corner (m-C)*
- *Minimum Bounding Cicle (MBC)* - O círculo circunscrito mínimo necessita de três parâmetros para determinar o centro e o raio do círculo.
- *Minimum bounding ellipse (MBE)* A elipse circunscrita mínima necessita de 5 parâmetros para seu armazenamento.

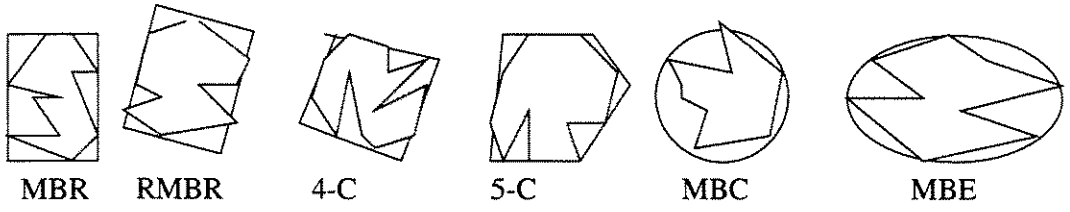


Figura 2.1: Aproximações conservativas convexas

2.2.2 Aproximações progressivas

Uma aproximação é dita progressiva se o conjunto de pontos da aproximação for um conjunto de pontos do objeto. Se as aproximações progressivas de dois objetos espaciais se interceptam então estes objetos também interceptam. Uma aproximação progressiva pode ser utilizada no lugar do objeto espacial que ela representa, por exemplo, com a finalidade de identificar objetos que atendem ao predicado de sobreposição. São exemplos de aproximações progressivas:

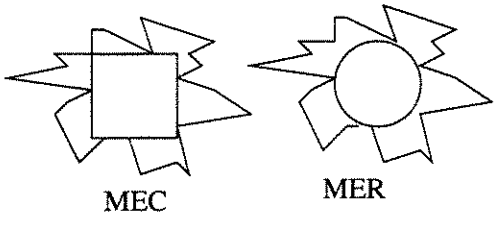


Figura 2.2: Aproximações progressivas

- *Maximum Enclosed Circle (MEC)* - O círculo inscrito máximo é calculado através do diagrama varonoi de seus lados sendo que um dos nós do diagrama é o centro do *MEC.*

- *Maximum Enclosed rectangle (MER)*- O retângulo inscrito máximo utiliza um parâmetro a menos que na representação *MEC*.

2.3 Predicados espaciais

Existem 3 tipos básicos de predicados espaciais:

- predicados direcionais (Ex: norte, sudeste, noroeste)
- predicados de distância (Ex: próximo, a 1 metro de)
- predicados topológicos (Ex: inclusão, contém, intercepta, toca)

Considerando como medida de desempenho o número de acessos a *buckets*, o trabalho [27] demonstra que o desempenho de consultas à janela, ou seja, consultas sobreposição, pode ser utilizado como uma medida global do desempenho do sistema. Em [27] foram analisados vários predicados espaciais bem como vários tipos de aproximações para os objetos espaciais. Devido a esse fato, o presente trabalho enfocará os predicados de sobreposição. O predicado de sobreposição pode ser assim definido: Sendo T um conjunto de objetos espaciais e r um retângulo encontrar todos objetos $o \in T$ *onde* $o \cap r \neq \emptyset$

2.4 Fases da junção

O processo de filtragem de objetos geométricos tem um grande impacto no processo de junções. A filtragem utiliza aproximações dos objetos e as consultas podem ser otimizadas pela aplicação de vários filtros. Os métodos de junção existentes são baseados na combinação de filtros e índices espaciais, sendo o índice utilizado para reduzir o custo da fase de filtragem e para minimizar o custo de recuperação dos objetos geográficos no disco. Os índices espaciais utilizam uma estrutura hierárquica de construção organizada de forma a possibilitar a busca de um conjunto de páginas de disco requisitadas, contendo um conjunto de objetos, com custo mínimo de disco E/S.

Um filtro é um preprocessador de uma operação complexa. A idéia principal de um filtro é reduzir o tamanho dos operandos. Com operandos menores, o custo de uma operação será menor, contudo o custo de utilização do filtro deve ser levado em consideração. Pode-se utilizar uma seqüência de filtros sendo que cada filtro na seqüência irá reduzir o tamanho do operando. Existem critérios [38] que podem ser utilizados para avaliar um modelo de filtragem tais como a qualidade da aproximação, os requisitos de armazenamento, o custo de um operação simples de teste, o custo de construção da aproximação e a insensibilidade à orientação. A utilização de uma aproximação de boa qualidade reduz

o número de pares candidatos, esta boa qualidade pode ser quantificada pela proporção entre a área da aproximação e a área do objeto. O critério de armazenamento será otimizado pela redução dos número de parâmetros que determinam a aproximação o que implica em uma redução do espaço necessário para armazenar a aproximação. O custo da operação de teste sobre o filtro deve ser bem menor que o custo de computar o predicado original sobre o objeto original de forma a justificar a utilização do filtro. A aproximação deve ter um baixo custo de construção e seria interessante que a rotação do objeto não causasse efeito no desempenho das consultas aplicadas a sua aproximação.

Os predicados aplicados aos objetos espaciais são complexos e tem um alto custo de aplicação. Consultas espaciais como as consultas de sobreposição devem ser otimizadas para que o mínimo delas sejam realmente calculadas. O desempenho da consultas espaciais melhora quando essas operações complexas são evitadas ou substituídas por operações mais simples.

Os resultados apresentados em [3] mostraram um desempenho satisfatório para o processamento de junção quando temos a implementação de três fases de filtragem, sendo os objetos espaciais armazenados junto de algumas das suas aproximações. A seguir temos o modelo *Multi-Step Spatial Query processor (MSQP)* utilizando três fases de filtragem [3]:

- A primeira fase denominada filtragem-*MBR* utiliza a aproximação *MBR* dos objetos espaciais . Se duas aproximações conservativas de objetos espaciais não se interceptam então isto implica que estes objetos não possuem interseção. Na primeira fase a aproximação *MBR* do objeto é utilizada para detectar pares que não fazem parte do conjunto solução, está fase retorna um conjunto de pares candidatos. Avaliar o predicado de sobreposição sobre aproximações *MBRs* de objetos espaciais tem um custo mais baixo que utilizar a representação geométrica exata do objeto, pois somente o *MBR* que possuí poucos parâmetros deve ser lido do disco e é possível aplicar técnicas de varredura de planos para simplificar a avaliação do predicado de sobreposição sobre os *MBRs*.

- Os pares candidatos provenientes da primeira fase de filtragem podem então ser avaliados em uma outra fase, a filtragem geométrica, para determinar quais destes candidatos devem ser descartados e também para determinar alguns pares que fazem parte do conjunto solução. A aproximação progressiva dos objetos espaciais é utilizada para determinar objetos que fazem parte do conjunto solução sem a necessidade de avaliar a representação geométrica exata do objeto. Experimentos [3] mostraram que a utilização do retângulo inscrito máximo detecta uma média de $\frac{1}{3}$ dos pares que fazem parte do conjunto solução. A utilização da aproximação conservativa envolvente mínima de 5 lados detecta aproximadamente $\frac{2}{3}$ dos pares candidatos que devem ser descartados [3].

- A terceira fase possui um alto custo e faz a avaliação do predicado de junção sobre a representação geométrica exata dos objetos candidatos provenientes da segunda fase. Para melhor o desempenho da terceira fase podem ser utilizadas técnicas de varredura de planos além de outras técnicas como a utilização da decomposição dos objetos em trapezóides e a aplicação do predicado sobre esses trapezóides [3].

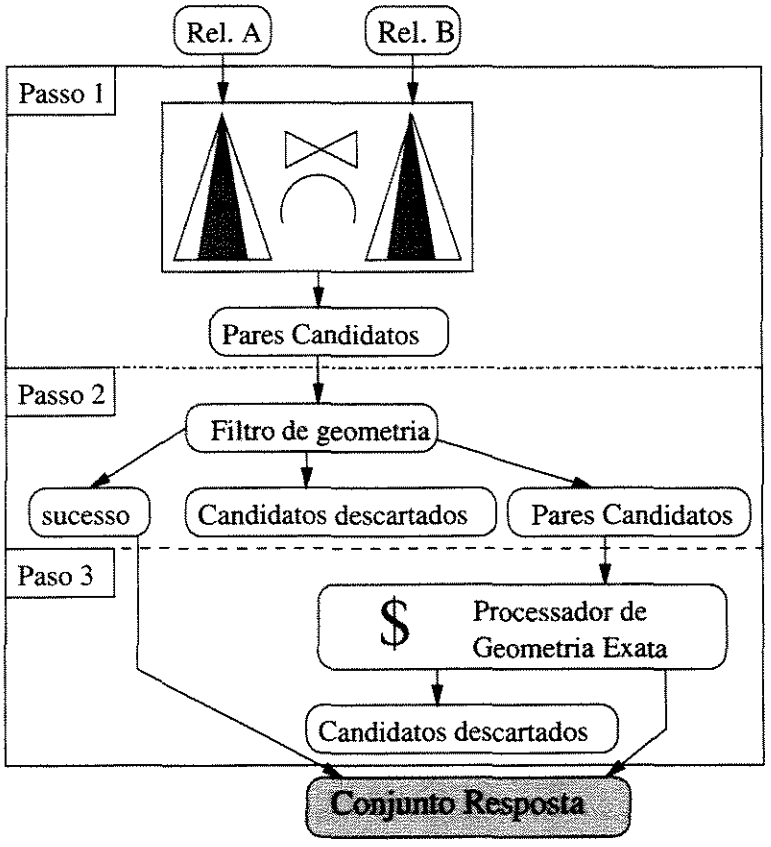


Figura 2.3: Fases da junção

2.4.1 Filtragem *MBR*

O modelo aqui proposto enfocará a primeira fase de filtragem *(MBR-join)* descrita à seguir. Considere dois conjuntos de objetos espaciais $A = \{a_1, .., a_n\}$ e $B = \{b_1, .., b_n\}$. Seja $Id()$ a função que associa um único identificador a cada objeto espacial e $Mbr()$ a função que computa o *MBR* destes objetos. A junção pode ser computada em um primeiro passo de filtragem, sendo que esse primeiro passo fornece um conjunto de pares candidatos a junção. O passo de *MBR-join* compõe-se na junção dos *MBRs* dos objetos geográficos e pode ser definido como a computação de todos os pares de $(Id(a_i), Id(b_j))$

com $Mbr(a_i) \cap Mbr(b_j) \neq \emptyset$.

2.5 Tipos de consulta

De um modo geral, temos um conjunto de tipos de consultas determinado pelo formato da janela de consulta. Se a janela for um ponto p, temos as consultas a ponto *(point queries)* nas quais busca-se todos os objetos do espaço que satisfazem um determinado predicado espacial com o ponto p. Caso a janela de consulta tenha forma retangular temos as *rectangle range queries* e busca-se pelos objetos que satisfazem um determinado predicado espacial em relação à janela. Se a janela possuir formato qualquer temos as consultas por região *(region queries)*, dependendo da complexidade desta janela temos um maior custo de avaliação do predicado espacial. O predicado espacial avaliado na consulta define também alguns subconjuntos dentro destes tipos de consulta. Esses predicados incluem o predicado de interseção , contém, adjacência, proximidade, direção dentre outros.

Neste trabalho analisaremos as consultas de janela retangular onde o predicado espacial sendo analisado é o predicado de interseção e a operação aplicada é a junção espacial *(intersection spatial join)*. Como nas junções os objetos tem que ser acessados várias vezes *(multi-scan queries)*, o tempo de execução não é linear no número de objetos. Já as *single-scan queries* fazem no máximo um acesso a um objeto e portanto, o tempo de execução é no máximo linear no número de objetos armazenados na relação correspondente. A seguir temos um exemplo de implementação da *single-scan queries* de busca por uma janela de consulta q.

```
01 WindowQuery(Rtree_Node N[i],window w)
02 FOR all s_k ∈ N[i] with s_k ∩ w ≠ ∅ DO
03   IF N[i] is a leaf node THEN
04     output(s_k)
05   ELSE /* nos intermediários */
06     ReadPage(s_k.ref)
07     WindowQuery(N[k],q)
```

2.6 Indexação

As junções espaciais podem ser classificadas em três categorias de acordo com a forma como os dados espaciais da junção estão organizados. Métodos de junção onde as relações não possuem índices, métodos de junções onde existem relações indexadas e não indexadas e métodos onde todas as relações estão indexadas. O presente estudo enfocará as junções

espaciais onde todas as relações da junção estão indexadas por um método de indexação baseado em árvores-R.

2.7 Indexação derivada de árvores multiárias

Os métodos de acessos multidimensionais *(MAMSs)* são estruturas de indexação projetadas para atuarem como um caminho otimizado aos dados espaciais com base em um conjunto de predicados espaciais definidos sobre os atributos. Um *MAM* é definido por meio de um conjunto de algoritmos de pesquisa e de alteração da estrutura de dados e por uma estrutura de dados que gerencia os objetos espaciais no disco [6].

O presente trabalho utiliza os métodos de indexação de objetos espaciais derivados de árvores multiárias. A familia das árvores-R foi utilizada para a indexação. As árvores-R são uma generalização das árvores-B (*B-trees*) para espaços K-dimensionais.

As árvores multiárias apresentam a vantagem de poder armazenar em um único nó muitas entradas referentes a vários filhos, com isso pode-se utilizar o tamanho de cada nó equivalente ao tamanho de uma página de disco. E dentro de cada nó as entradas mantém uma correlação de proximidade espacial. Uma página de disco corresponde a um nó e acessa um conjunto de entradas que apresentam forte correlação espacial, facilitando o processo de pesquisa e de execussão de consultas espaciais e o gerenciamento dos objetos espaciais no disco. Como as árvores multiárias são construídas de baixo para cima, todos os nó folhas aparecem no mesmo nível da árvore, isto é, a árvore é balanceada. Além disso, não é necessário reorganizar periódicamente os dados na estrutura, pois nas árvores multiárias o desempenho das estruturas de indexação não se degrada devido às constantes atualizações nos dados.

Apresentaremos aqui apenas, uma visão geral relativa as estruturas de indexação espacial baseadas nas árvores multiarias, mais especificamente as árvores-R [14], para maiores detalhes de implementação verificar os trabalhos [12, 5, 9, 6].

2.8 Árvores da família R

As árvores multiárias da família R são uma generalização das árvores-B para espaços k-dimensionais e armazenam os *MBRs* dos objetos espaciais através de um estrutura hierarquica de retângulos, sendo que as referências e as aproximações dos objetos espaciais propriamente ditas estão no nível das folhas.

2.8.1 Árvore R

A árvore-R possui dois tipos básicos de nós:

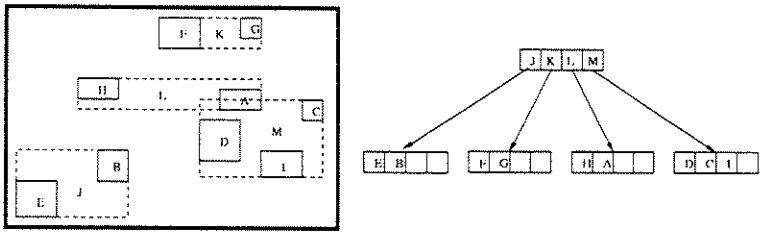


Figura 2.4: Uma árvore-R e os retângulos indexados. Os *MBRs* de um mesmo nível podem se interceptar.

- nós folhas, que armazenam informações sobre os objetos indexados na forma (*identificador, MBR*), onde o primeiro campo é o identificador do objeto e o segundo campo é o *MBR* do objeto
- nós não-folha, que contém informações da forma (*ponteiro, MBR*), onde o primeiro campo é um ponteiro para um nó de um nível inferior na hierarquia e *MBR* é o retângulo envolvente mínimo que envolve todos os *MBRs* dos nós subordinados.

Cada nó é armazenado em uma página de disco, cuja capacidade máxima é M entradas do tipo (*ponteiro, MBR*) ou (*identificador, MBR*). O número mínimo de entradas em um nó é dado por $m \leq \frac{M}{2}$. Definidos estes dois valores, um nó deve respeitar as seguintes características:

- um nó não deve possuir mais do que M e menos do que m entradas, a menos que seja raiz.
- para cada registro *(identificador, MBR)* em um nó folha, *MBR* é o menor retângulo que espacialmente contém o objeto de dados representado pelo registro.
- todo nó não-folha possui entre m e M filhos, a menos que seja raiz.
- para cada entrada *(ponteiro, MBR)* em um nó não-folha, *MBR* é o menor retângulo que espacialmente contém os retângulos do nó filho apontado por (*ponteiro*).
- o nó raiz possui pelo menos dois filhos a menos que seja folha.
- todas as folhas aparecem no mesmo nível.

Os novos objetos são inseridos nas folhas como nas árvores-B$^+$. Para inserir um novo objeto, percorre-se a estrutura da árvore a partir da raiz e segue-se o ponteiro cujo *MBR* associado necessite do menor aumento de área para conter o *MBR* do objeto. Isto pode acarretar um *overflow*, ou seja, esta folha pode já ter atingido o número máximo de

entradas permitido. Neste caso é aplicada a folha uma operação de *split* onde uma nova folha é criada e as entradas da entrada que sofre o *split* mais a nova entrada são divididas entre a follha que sofreu *split* e a folha que foi criada. A operação de *split* pode se propagar até os níveis superiores da estrutura, atingindo até mesmo a raiz, uma vez que a criação da nova folha implica na criação de uma nova entrada no nó imediatamente superior a ela.

Consultas em uma árvore-R são realizadas avaliando-se predicados entre a janela de consulta e os *MBRs* armazenados nos nós da árvore. Para se determinar quais objetos interceptam a janela de consulta, percorre-se a árvore a partir da raiz, descendo até as folhas, de modo similar a uma árvore-B. Se o nó não é uma folha, verificam-se todas as entradas em busca daquelas cujo *MBR* intercepta a janela de consulta e, para toda entrada nessa condição, verifica-se o nó associado. Caso o nó seja folha, as entradas que interceptam a janela de consulta são computadas no conjunto resposta.

A remoção de uma entrada da árvore inicia-se com a determinação da folha onde se encontra a mesma, o que é feito chamando-se o procedimento de consulta. Então elimina-se a entrada. Após eliminar a entrada, se o nó onde ela se encontrava contiver m ou mais entradas, os *MBRs* dos nós pertencentes ao caminho percorrido até a folha devem ser verificados e se necessário, ajustados. Caso o número de entradas seja menor que o exigido (evento conhecido por *underflow*), as entradas restantes são atribuídas a um conjunto Q para serem inseridas mais tarde.

2.8.2 árvore-R$^+$

A árvore-R$^+$ [33] é uma estrutura de indexação que procura dividir o espaço sem sobreposição dos *MBRs* de um mesmo nível. Com isso, busca eliminar a interseção de retângulos existente na árvores-R e, em conseqüência, melhorar o desempenho de consultas. Para isso, o *MBR* de um objeto é decomposto em uma uma coleção de sub-retângulos disjuntos, cuja união é o próprio *MBR* do objeto, sempre que isso seja necessário para que os *MBRs* dos nós superiores não se interceptem.

Sua estrutura é a mesma das árvores-R, mas as regras de construção são diferentes:

- para cada entrada (*ponteiro, MBR*) em um nó não-folha, a sub-árvore apontada por *ponteiro* contém um retângulo R se, e somente se, R for totalmente sobreposto por *MBR* . Se R está em um nó folha, R pode apenas interceptar *MBR*.

- para quaisquer duas entradas ($ponteiro_1$, MBR_1) e ($ponteiro_2$, MBR_2) de um nó não-folha, a interseção entre MBR_1 e MBR_2 é nula.

- a raiz possui pelo menos dois filhos, a menos que seja folha;

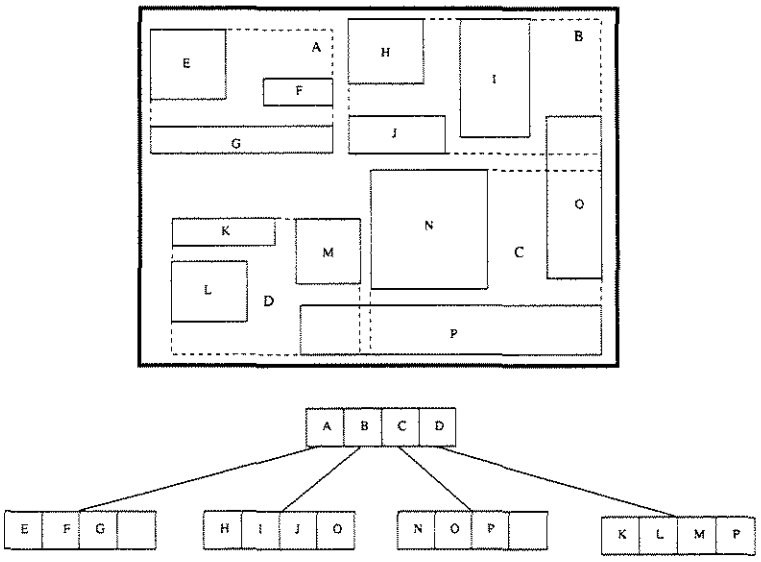


Figura 2.5: Os retângulos indexados por uma árvore-R^+.

- todas as folhas se encontram no mesmo nível.

A Figura 2.5 mostra os pontos e retângulos indexados por uma árvore-R^+. Note que o retângulo obj_7 foi representado nas folhas R_2 e R_3 para permitir que os retângulos envolventes fossem disjuntos. A operação de inserção para a árvore-R^+ é executada percorrendo-se a estrutura da árvore a partir da raiz. Todos os nós não folha cujos *MBRs* interseptem o *MBR* do objeto tem suas sub-árvores percorridas. Nas árvore-R^+ as entradas podem se interseptar gerando buscas mais caras. A busca é realizada através da decomposição do espaço em sub-regiões disjuntas e percorrendo-se cada uma até que os objetos sejam encontrados nas folhas. A operação de remoção é realizada removendo-se a entrada de todos os nós folhas onde ela estiver presente.

2.8.3 Árvore-R*

A árvore-R* [2] procura melhorar o agrupamento de retângulos feito pela árvore-R. Possui a mesma estrutura da árvore-R, e os mesmos algoritmos de consulta e exclusão. A principal diferença entre elas está no algoritmo de inserção. Enquanto que na árvore-R a heurística de otimização baseia-se apenas na área do retângulo, a R* baseia-se em vários fatores:

- a *sobreposição* entre retângulos dos nós não-folha. A diminuição da sobreposição melhora o desempenho, uma vez que diminui o número de caminhos possíveis a serem percorridos nas consultas;

- a *ocupação* dos nós. O aumento da ocupação diminui o custo das consultas, pois reduz a altura da árvore;

- o *perímetro* dos retângulos. Assumindo uma área fixa, o objeto com menor perímetro é um quadrado. Procura-se então minimizar o perímetro ao invés da área, buscando gerar *MBRs* mais próximos de quadrados. Uma vez que quadrados tendem a ser agrupados mais facilmente, o agrupamento de *MBRs* de determinado nível resultaria em *MBRs* menores nos níveis superiores.

Os fatores citados acima estão concentrados na rotina que escolhe a folha onde um retângulo deve ser armazenado e na que trata do *overflow* na inserção de uma entrada.

2.8.4 Hilbert R-tree

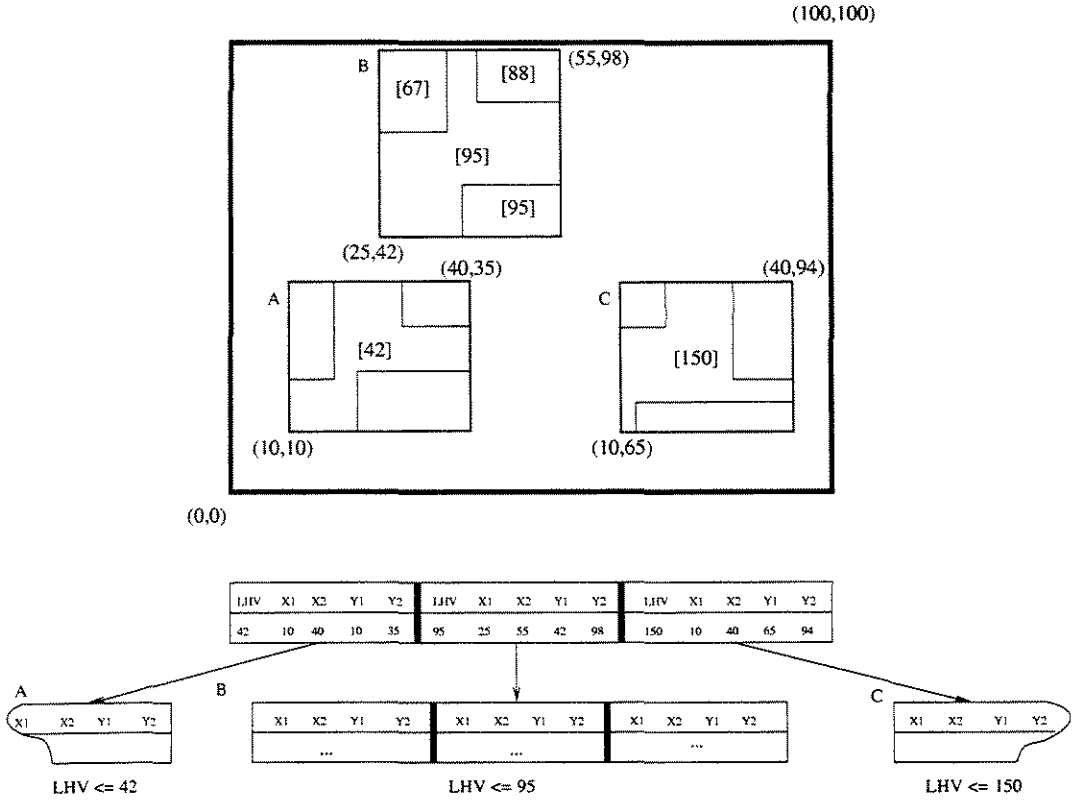


Figura 2.6: Os retângulos indexados de uma Hilbert R-tree (a). Entre colchetes estão as coordenadas da curva de Hilbert enquanto que entre parênteses estão as coordenadas no espaço. A árvore é apresentada na Figura (b).

A Hilbert R-tree [19] é uma estrutura de indexação que tem como característica diferencial procurar adiar ao máximo a execução das rotinas de *split*. A idéia é criar uma vizinhança s de nós através da ordenação de nós baseado-se na *space-filling curve* de Hilbert. Assim, se um nó tiver sua capacidade ultrapassada, ao invés de se executar o *split*, procura-se passar algumas de suas entradas aos seus $s-1$ vizinhos, adiando o *split* ao máximo. O *split* só é executado caso todos os nós de s estiverem cheios. Esta característica da *Hilbert R-tree* permite que, ajustando-se s, a ocupação dos nós fique muito próxima de 100%. A *Hilbert R-tree* possui a seguinte estrutura:

- nós folha, contendo no máximo C_f entradas do tipo *(identificador, MBR)*, onde C_f é a capacidade da folha, identificador é um ponteiro para a descrição do objeto e *MBR* é seu retângulo envolvente mínimo;
- nós não-folha, contendo no máximo C_n entradas do tipo *(ponteiro, MBR, LHV)*, onde C_n é a capacidade do nó, ponteiro indica um nó filho, *MBR* é o retângulo envolvente mínimo da sub-árvore apontada por *ponteiro* e LHV é o maior valor na curva de Hilbert entre os retângulos de dados. As coordenadas dos *MBRs* dos nós não-folha nunca são calculadas. Os valores LHV são calculados tomando-se os centros dos *MBRs*.

2.9 Junções Espaciais

Um exemplo de junção espacial seria uma consulta do tipo: Dado um cunjunto de áreas onde se encontra o foco do mosquito da dengue, quais os postos de saúde que pertencem a essas áreas?

O processamento eficiente de uma junção espacial é extremamente importante, já que seu tempo de execução é superior a linear no número de objetos espaciais das relações participantes, e esse número de objetos pode ser muito grande. A junção espacial é uma das operações mais caras que um *GIS* suporta.

Uma junção espacial entre as relações A e B é uma operação que constrói pares de tuplas a partir de $A \times B$, cujo atributo espacial satisfaz um predicado espacial. Há vários predicados espaciais, dentre os quais podemos citar "interseção", "contém", "à distância de", "em direção a", sendo que o predicado mais comum é interseção [12].

A junção espacial pode ser assim definida: Dados dois conjuntos de objetos espaciais A e B e um predicado espacial Θ, determine todos os pares de objetos $(o, o') \in A \times B$ onde $\Theta(o.G, o'.G)$ é verdadeiro. O predicado espacial pode ser definido a partir de um relacionamento topológico (por exemplo, intersecta, contém, é adjacente a), a partir de um relacionamento métrico (como exemplo, *k-nearest neighbor* e [distância $\{=, \leq, <, >, \geq\} q$]), ou de relacionamento direcional (tal como, ao norte de, ao sul de, a leste de e a oeste

de). Desta forma, percebe-se que o uso de diferentes tipos de relacionamento conduz a subtipos específicos de junção espacial. A definição formal genérica da junção espacial é:

$$A \bowtie_{\Theta} B = \{(o, o') | \, o \in A \wedge o' \in B \wedge \Theta(o.G, o'.G)\} \tag{2.1}$$

Para realizar a operação de junção espacial é conveniente utilizar as pesquisas se ambas as relações forem indexadas. As pesquisas percorrem de forma organizada as estruturas de indexação de forma a reduzir o número de acessos ao disco para otimizar a operação de junção. Nas seções seguintes estão descritos dois tipos básicos de pesquisa para a árvore de indexação: a busca em profundidade na seção 2.10 e a busca em largura na seção 2.11.

2.10 Busca em profundidade

Considere dois conjuntos de objetos espaciais aproximados por seus *MBRs* e indexados por um estrutura baseada na árvore-R e uma junção espacial para o predicado sobreposição a ser executada sobre esses dados. O trabalho [4] propõe a busca em profundidade para executar essa junção e o emprego das técnica de otimização local para melhorar o desempenho da junção. Para redução do tempo de UCP propõem-se as técnicas de restrição do espaço de busca e varredura de planos. Para diminuir o número de acessos a disco propõe-se a técnica de fixação das páginas de *buffer* além da utilização de um *buffer* gerenciado pela política *LRU*.

Quando temos um conjunto de objetos indexados por uma estrutura baseada na árvore-R a busca em profundidade apresenta a vantagem de executar uma poda durante o processo de junção. Essa poda baseia-se no fato de que se em um dado nível da árvore um nó não satisfaz ao predicado da junção, então os nós filhos deste nó também não satisfarão o predicado. Portanto não é necessário avaliar o predicado para os filhos o que economiza acessos a disco e tempo de UCP. A seguir temos uma implementação para o algoritmo de busca em profundidade.

```
PROCEDURE RtreeJoin(RtreeNode R, S)
/* R, S são duas árvores R de mesma altura hR = hS */
01  FOR all E_i ∈ R DO /* restrição do espaço de busca em R */
02    IF E_i.rect ∩ S.rect == ∅ THEN R = R − {E_i};
03  FOR all E_j ∈ S DO /* restrição do espaço de busca em S */
04    IF E_j.rect ∩ R.rect == ∅ THEN S = S − {E_j};
05  Sort(R); Sort(S);
06  SorteIntersectionTest(R, S, Seq); /* Varredura de planos */
07  FOR i = 1 TO ||Seq|| DO
```

```
08    (E_R, E_S) = Seq[i];
09  IF R is a leaf page THEN /* S também é uma página folha */
10    output(E_R, E_S);
11  ELSE
12    ReadPage(E_R.ref); ReadPage(E_S.ref);
13    RtreeJoin(E_R.ref, E_S.ref);
```

2.10.1 Tempo de UCP

Restrição do espaço de busca

A técnica de restrição do espaço de busca excluí as entradas cujos retângulos não tem interseção com o retângulo que engloba o outro nó, antes da interseção ser efetivada entre os retângulos das entradas de ambos os nós.

Considerando dois nós não folha E_R e E_S de duas árvores-R, R_1 e R_2, para os quais está sendo verificada uma junção de sobreposição entre os elementos das árvores R_1 e R_2. Considerando o predicado de sobreposição avaliado como verdadeiro para os nós E_R e E_S, ou seja, $E_R.rect \cap E_S.rect \neq \emptyset$, temos que somente as entradas de $E_1.ref$ e $E_2.ref$ que interceptam o retângulo de interseção $E_1.rect \cap E_2.rect$ podem ter uma intercessão comum. Portanto uma busca linear dentro de cada um dos dois nós é utilizada para selecionar apenas os nós que pertencem à interseção dos dois retângulos, e a junção só precisa ser avaliada para esses nós.

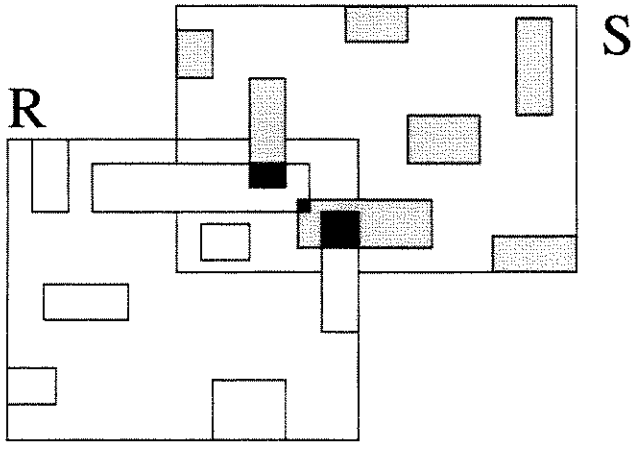


Figura 2.7: Restrição do espaço de busca

Varredura de planos

Um retângulo r_i é caracterizado por seu canto inferior esquerdo $(r_i.xl, r_i.yl)$ e por seu canto superior direito $(r_i.xu, r_i.yu)$. Dadas duas seqüências de retângulos $Rseq =< r_1, .., r_n >$ com n retângulos e $Sseq =< s_1, .., s_m >$ de m retângulos. Seja $\pi_X(r_i)$ e $\pi_Y(r_i)$ as projeções de r_i nos eixos X e Y e seja $\pi_X(s_i)$ e $\pi_Y(s_i)$ as projeções de s_i nos eixos X e Y respectivamente. Primeiramente cada uma das seqüências é ordenada separadamente em relação a um dos eixos. Considere uma ordenação inicial em relação ao eixo X. Diz-se que uma seqüência qualquer $Useq =< u_1, .., u_w >$ está ordenada em relação ao eixo X, se $u_i.xl \leq u_{i+1}.xl, 1 \leq i < w$. Portanto, após a ordenação das seqüências $Rseq$ e $Sseq$ teremos $r_i.xl \leq r_{i+1}.xl, 1 \leq i < n$ e $s_i.xl \leq s_{i+1}.xl, 1 \leq i < m$.

Uma linha denominada linha de varredura *(sweep line)* perpendicular ao eixo das projeções, ou seja, perpendicular ao eixo X, move-se no conjunto $Rseq \cup Sseq$ a partir do retângulo t com o menor valor de lx. Se o retângulo t pertence à seqüência $Rseq$, percorre-se seqüencialmente $Sseq$ começando por seu primeiro retângulo até encontrar um retângulo s_h cujo valor de lx for maior que $t.xu$. Com isso temos que o intervalo $\pi_X(t)$ interseciona o intervalo $\pi_X(s_j)$, $\forall j$ em $1 \leq j < h$. Se o intervalo $\pi_Y(t)$ interceptar o intervalo $\pi_Y(s_j)$ então o retângulo t intercepta o retângulo s_j. Se o retângulo t está em $Sseq$, $Rseq$ é percorrida de forma análoga. O retângulo t é marcado como processado. Então a linha de varredura é movida para o próximo retângulo do conjunto $Rseq \cup Sseq$ com o menor valor de xl que não esteja marcado. O processo continua até que todos os retângulos de $Rseq \cup Sseq$ sejam marcados completando o processo de determinar as interseções. A seguir temos o algoritmo *SortedIntersectionTest* que implementa a varredura de planos e tem complexidade $O(||Rseq|| + ||Sseq|| + k_x)$, onde k_x representa o número de pares de interseção dos intervalos criados pelas projeções dos retângulos de $Rseq$ e $Sseq$ sobre o eixo X.

```
SortedIntersectionTest(Rseq,Sseq: SEQUENCE of Retangle;
                       VAR Output: SEQUENCE OF PAIR OF Rectangle);
/* Rseq e Sseq estão ordenados */
01  Output:=<>; i:=1; j:=1;
02  WHILE  (i ≤ ||Rseq||) ∧ (j ≤ ||Sseq||) DO
03    IF r_i.xl < s_j.xl THEN /* u = r_i */
04      InternalLoop(r_i, j, Sseq, Output)
05      i:=i+1;
06    ELSE  /* u = s_j */
07      InternalLoop(s_j, i, Rseq, Output)
08      j:=j+1;
09    END
```

```
10  END
11 END SortedIntersectionTest
```

```
InternalLoop(t: Rectangle; unmarked: CARDINAL;
                           Sseq: SEQUENCE of Retangle;
                           VAR Output: SEQUENCE OF PAIR OF Rectangle);
01  k:= unmarked;
02  WHILE  (k ≤ ||Sseq||) ∧ (s_k.xl ≤ t.xu) DO    /* Interseção-X */
03    IF  (t.yl < s_k.yu) ∧ (t.yu > s_k.yl)   THEN /* Interseção-Y */
04      Append(Output, (t.s_k))
05    END;
06    k:=k+1;
07  END
11 END InternalLoop;
```

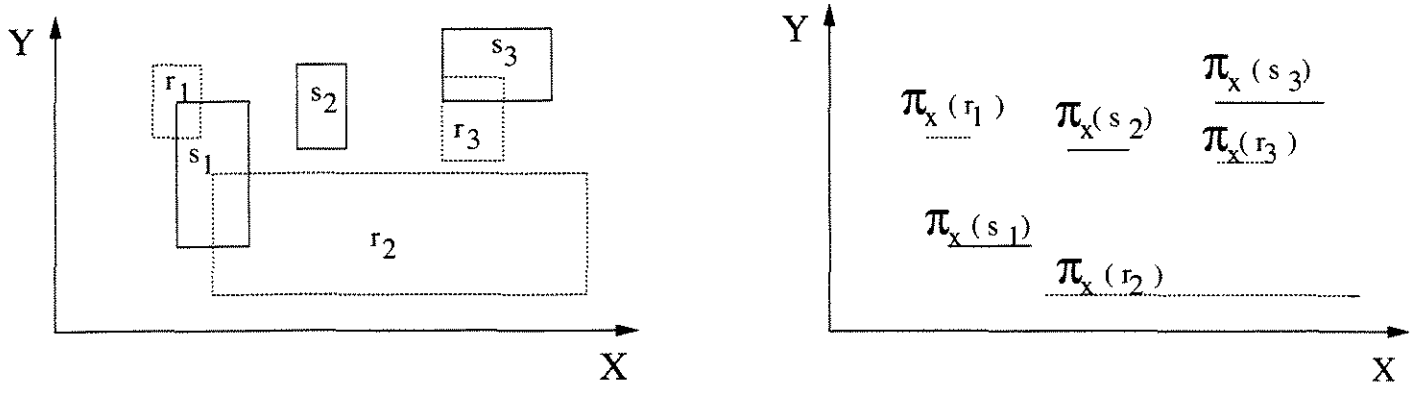


Figura 2.8: Dois conjuntos de retângulos e suas projeções no eixo X

A figura 2.8 representa o seguinte conjunto de testes de interseção:
$t = r_1 : r_1 \longleftrightarrow s_1$
$t = s_1 : s_1 \longleftrightarrow r_2$
$t = r_2 : r_2 \longleftrightarrow s_2, r_2 \longleftrightarrow s_3$
$t = s_2 : -$
$t = r_3 : r_3 \longleftrightarrow s_3$

2.10.2 Tempo de E/S

Fixação das páginas de *buffer*

Para diminuir o tempo de E/S é necessário reduzir o número de acessos a disco, portanto as páginas que são utilizadas durante a junção devem estar no *buffer-pool*. O problema de E/S caracteriza-se pela definição de um escalonamento de leitura para as páginas utilizadas na junção. Em [29] mostrou-se que o escalonamento destas páginas é um problema NP-difícil, portanto são utilizadas heurísticas como uma solução aproximada para o problema. Seja um *buffer* compartilhado gerenciado pela política *LRU* e uma partição deste *buffer* sendo reservada para a computação de junções espaciais. No processamento da junção um nó pode ser consultado mais de uma vez, dependendo do tamanho e do gerenciamento do *buffer* estas novas consultas podem levar a *page fauls* no *buffer* implicando em acessos ao disco. Para evitar que páginas que contém nós que serão usados novamente na computação da junção sejam retiradas do *buffer*, além da política *LRU*, outras técnicas baseadas em heurística que prevêem os nós que serão utilizados no futuro são empregadas para fixar as páginas que contém esse nó no *buffer*. Um possível escalonamento para as páginas seria utilizar a seqüência proveniente da varredura de planos, esta seqüência apresenta a vantagem de preservar parte da localidade espacial dos objetos escalonados. A técnica de fixação das páginas de *buffer* utiliza informações provenientes da varredura de planos como sua heurística de definição de páginas a serem fixadas no *buffer*. Durante o processamento da varredura de planos, um contador é guardado para cada nó que fizer parte da varredura, esse contador é inicializado em zero e a cada sobreposição determinada pela varredura de planos, os nós que fazem parte desta sobreposição recebem um incremento de um nos seus contadores. As páginas que contiverem o nó com maior contador são fixadas no *buffer* durante a junção. Então a junção é realizada entre a página fixada no *buffer*, cujo retângulo correspondente tenha grau máximo, com todos os outros retângulos. Então o processo se repete para o próximo nó com grau máximo. O próximo nó a ser escolhido tem por base a ordem de escalonamento da varredura de planos.

2.11 Busca em Largura

Dados dois conjuntos de dados espaciais aproximados pelos seus *MBRs* e sendo indexados por duas estruturas baseadas na árvore-R e uma junção espacial para avaliar o predicado de sobreposição para esse conjunto de dados, a busca em largura para junção espacial *(Breadth-First R-tree join BFRJ)* é empregada para computar essa junção. O algoritmo de busca percorre ambas as árvores-R, varrendo totalmente cada um dos níveis da árvore. A *BFRJ* apresenta vantagens quando comparada a busca em profundidade comumente empregada para avaliar a junção. A busca em largura, por acessar todos os nós de um nível

de um árvore em seqüência, pode aplicar otimizações globais que diminuem o número de acessos a disco para a junção. Essas otimizações baseiam-se em um índice intermediário de junção *(Intermediate Join Index IJI)* que é construído para cada nível das árvores durante a junção. As estratégias aplicadas ao *IJI* incluem a ordenação do índice, o gerenciamento de memória e de *buffer* baseados no índice. Experimentos [17] utilizando a árvore-R estástica *(packed R-tree)* mostram que em comparação a busca em profundidade, a busca em largura apresenta um desempenho 50% superior quanto ao número de acessos a disco para um espaço de *buffer* de tamanho médio ou grande reservado para computar a junção. Como os banco de dados modernos tendem a possuir um *buffer* de tamanho grande para computar as junções eles tem acesso a pelo menos um *buffer* de tamanho médio. A busca em largura também incorpora as técnicas de otimização local de restrição do espaço de busca (seção 2.10.1) e varredura de planos (seção 2.10.1) que melhoram a eficiência da computação no processo de junção para cada par de nós.

2.11.1 Algoritmo para BFRJ

A junção implementada através de uma busca em uma estrutura indexada por árvores-R tem a vantagem de poder realizar uma poda durante a busca. Isto diminui o número de nós visitados e baseia-se no fato da árvore-R preservar a proximidade espacial entre os objetos em um mesmo nó e porque existe uma hierarquia de granularidade relacionando os pais com granularidade alta aos seus respectivos nós filhos com granularidade mais baixa na árvore. Se dois nós pais de duas árvores distintas não tem interseção então, por construção, os seus filhos também não terão interseção, isto significa que este predicado não precisará ser avaliado para os filhos, nisto consiste a poda.
O *BFRJ* primeiramente checa a interseção para os *MBRs* dos nós raízes das duas árvores nR^0 e nS^0 e armazena as interseções avaliadas $< oidR^0, oisS^0 >$ no índice intermediário de junção do nível zero IJI_0. Para cada dois nós referenciados no IJI_0, o algoritmo que implementa o *BFRJ* busca os dois nós referenciados como R e S e executa uma junção espacial com esses pares de nós sendo o resultado desta junção armazenado em um novo índice de junção intermediário. Quando conclui-se a avaliação do IJI_0 este índice é descartado e começa o mesmo tipo de avaliação de junção para o IJI_1 e assim sucessivamente.
Considerando-se duas árvores-R de indexação R e S de mesma altura, abaixo temos o código do algoritmo *BFRJ*. O procedimento *Node_Pair_Join_Same_Level()* usa dois nós das árvores R e S como entrada e computa a junção espacial entre os *MBRs* armazenados nos dois nós.

```
PROCEDURE BFRJ-Same-Height(R, S)
// R, S são duas árvores R de mesma altura hR = hS
```

```
DATA STRUCTURES: set IJI[hR]:=0;
// IJI[i] é o índice intermediário de junção criado no nível i
//Faz a junção dos nós da raiz
01  IJI[0]:= Node_Pair_Join_Same_Level(nR^0, nS^0);
02  integer i:=0;
03  while i < hR - 1 do
04  ∀ < oidR^i, oisS^i > ∈ IJI[i] do
05      IJI[i+1] = IJI[i+1] ∪ Node_Pair_Same_level(nR^i, nS^i);
06    end do
07    i:= i + 1; // desce um nível
08  end while
09  output IJI[i]; //A saída é IJI[i]
```

Para $hR < hS$, isto é, árvores com altura diferente aplica-se o algoritmo abaixo. Até a árvore de altura menor atingir o nível das folhas, o algoritmo se comporta de maneira idêntica ao algoritmo descrito *BFRJ-Same-Height*. Depois de atingido o nível $hR - 1$ o algoritmo *BFRJ* permanece no mesmo nível para a árvore de menor altura R mas continua a descer os níveis para a outra árvore, até atingir o nível das folhas para a árvore de maior altura S. No algoritmo apresentado abaixo de *BFRJ* para árvores de altura diferente, o procedimento *Node_Pair_Join()* sofre uma modificação de comportamento se comparado ao *Node_Pair_Join_Same_Level()*. O *Node_Pair_Join()* quando verifica o predicado para nós folhas da árvore de menor altura com nó não folhas da árvore de maior altura, avalia o predicado entre as aproximações dos objetos *MBRs* que estão em R com as aproximação de maior granularidade compostas pelo *MBR* que cobre os *MBRs* dos nós filhos do nó em S.

```
PROCEDURE BFRJ(R, S)
DATA STRUCTURES: set IJI[max(hR, hS)]:=0;
// IJI[i] é o índice intermediário de junção criado no nível i
01  IJI[0]:= Node_Pair_Join(nR^0, nSsp0); //faz a junção dos nós da raiz
02  integer i:= r:= s:= 0;
03  while r < hR - 1 or s < hS - 1 do
04  ∀ < oidR^i, oisS^i > ∈ IJI[i] do
05      IJI[i+1] = IJI[i+1] ∪ Node_Pair_Same_level(nR^r, nS^s);
06    end do
07  if r :≠ hR − 1 then
08    r:= r + 1; //desce um nível se ainda não atingiu as folhas
09  end if
10  if s :≠ hS − 1 then
```

```
11    s:= s + 1; //desce um nível se ainda não atingiu o nível das folhas
12  end if
13    i:= i + 1; // desce um nível
14  end while
15  output IJI[i]; //A saída é IJI[i]
```

O IJI_i criado para cada nível i da busca pode ser usado para computar a junção para o nível $i+1$ junto com os nós do nível $i+1$ das árvores. O *IJI* fornece informações globais sobre a ordem de acesso aos nós e o número de vezes que cada nó foi acessado. Utilizando essas informações é possível aplicar-se técnicas de otimização global a busca, como a ordenação dos índices e o gerenciamento de memória e *buffer*.

2.11.2 Ordenação dos índices intermediários

Supondo que o *MBR* de um nó de R nR_i^l tenha interseção com $k > 1$ diferentes nós de S para o nível l. O *IJI* construído um nível acima $(l-1)$ conterá k vezes o *ID* do nR_i^l. Logo se tivermos um grande distanciamento nas ocorrências das aparições de um mesmo *ID* no *IJI* poderemos ter muitos *page fault* no *buffer*. A técnica de ordenação dos índices tem como princípio à manutenção de *ID* iguais o mais próximo possível para evitar que o nó saia do *buffer* por não estar sendo utilizado, a ordenação diminui as *page fauls* no *buffer* melhorando o desempenho. Porém cada par presente no IJI é composto de um ID para cada uma das duas árvores, se a ordenação ocorrer baseada em um ID de uma das árvores apenas isso não garante que a ordenação estará boa também para os IDs da outra árvore. Dado um *IJI* composto de pares da forma $< oidR_i, oidS_i >$ existem algumas políticas que definem um modelamento para essas ordenações:

OrdNo: Política de custo zero, não existe ordenação. Os pares no *IJI* permanecem na ordem em que eles foram construídos. Com a realização do *plane sweep* já realizou uma correlação simples entre os nós, eles já estão em uma ordenação fraca por localidade dentro do *IJI*.

OrdOne: A ordenação é feita baseando-se apenas nos *IDs* de uma das árvores. Uma ordenação de *IDs* para uma das árvores não garante um boa ordenação para os *IDs* da outra árvore.

OrdSum: Calcula o centro x dos valores das coordenadas dos *MBRs* para os *oidR* e *oidS*:
$CX_{oidR} = (lx_{oidR} + hx_{oidR})/2$
$CX_{oidS} = (lx_{oidS} + hx_{oidS})/2$
A ordenação é feita baseando-se na soma de CX_{oidR} e CX_{oidR}, isto é, na soma dos centros:
$sortKey = (lx_{oidR} + hx_{oidR})/2 + (lx_{oidS} + hx_{oidS})/2$

OrdeCen: Cria um *MBR* que engloba os *MBRs* de *oidR* e *oidS*. A ordenação baseia-se nos

valores da coordenada x do ponto central do *MBR* criado. Seja lx_{min} o menor lx dentre os *MBRs* de *oidR* e *oidS* e seja hx_{max} o maior hx entre os *MBRs* de *oidR* e *oidS*:
$sortKey = (lx_{min} + hx_{max})/2$
OrdHil: É uma ordenação semelhante a *OrdCen*, a ordenação baseia-se no valor da coordenada x do *MBR* que engloba os *MBRs* de *oidR* e *oidS*. *OrdHill* faz a ordenação baseando-se no valor curva de *Hilbert* para esses pontos centrais.

2.11.3 Gerenciamento de Memória

O *IJI* atingi seu tamanho máximo quando a busca em largura constrói o IJI_i para o nível i anterior ao nível folha da árvore mais alta. Dependendo do tamanho máximo do índice intermediário IJI_{max} decide-se por armazenar o *IJI* no disco ou no *buffer*. Para que o *IJI* seja armazenado no *buffer* seu tamanho máximo tem que ser menor que o tamanho do *buffer*, neste caso sobra menos espaço no *buffer* para guardar os nós e computar a junção. Se o IJI_{max} ficar muito grande a ponto de não caber no *buffer* é necessário armazená-lo no disco o que gera um custo adicional de acessar o disco. O *IJI* é ordenado no *buffer*. A ordenação do *IJI* ocorre depois de ter sido processada a junção para um nível da árvore e antes de se começar a processar a junção para o próximo nível. Por isso, o espaço de *buffer* destinado ao processo de junção pode ser totalmente usado para a ordenação do *IJI*. Se o *IJI* for armazenado no disco ele estará ordenado no disco, portanto somente uma página de *buffer* é necessária para trazer parte do índice para o *buffer* já que ele é escrito seqüencialmente. Considerando um *buffer* gerenciado pela política LRU de gerenciamento de páginas, tem-se que a página que contém o *IJI* por estar sendo continuamente usada pelo processo de junção não será retirada do *buffer*.

2.11.4 Gerenciamento de *buffer*

Como não é possível definir uma ordenação perfeita para os nós relativos ás duas árvores no *IJI*, podem ocorrer *page fauls* na busca a um par do *IJI* durante a junção. Para diminuir o número dessas *page fauls* o gerenciador do *buffer* deve ser capaz de prever quais páginas do *buffer* serão utilizadas no futuro. Esta previsão é baseada em um contador que é inicializado em zero para cada nó de ambas as árvores quando inicia-se a criação do *IJI* para um dado nível. Quando o *IJI* está sendo criado, o contador referente a cada nó adicionado ao *IJI* é incrementado. As páginas dos nós que possuírem contador maior que zero são fixadas no *buffer*. Durante a junção os nós do par do *IJI* que estiverem sendo computados receberão um decremento de um. A página que possuir todos os nós com contador zero pode ser liberada para substituição.

2.12 Junções espaciais M

Uma maneira de processar o *Multi-way R-tree Join* é aplicando-se uma seqüência de *2-way R-tree Joins* criando resultados intermediários. É possível também avaliar todas as relações da junção ao mesmo tempo *(Multi-way spatial joins)*. A seguir apresentamos alguns possíveis processamentos para as *Multi-way spatial joins*.

2.12.1 Modelo baseado no algoritmo MFC

O trabalho [30] propõe um mapeamento da junção espacial M para o problema de satisfação de restrição *(Constraint Satisfaction Problem CSP)*. Os algoritmos do CSP são utilizados para resolver o problema da junção M. Um problema de satisfação de restrição é definido por:

- Um conjunto de n variáveis $v_1, ..., v_n$. Considere o problema de junção espacial M assim descrito: encontrar todas as cidades cortadas por uma estrada que também corta um área de preservação ecológica. As variáveis do CSP poderiam ser mapeadas para cidade, estrada e área de preservação ecológica.

- Sendo N_i a cardinalidade de v_i, tem-se o domínio finito $D_i = \{u_{i,1}, .., u_{i,N_i}\}$ definido para cada variável v_i. O conjunto de nomes das cidades pode ser mapeado como a domínio da variável cidade no problema *CSP*.

- Para cada par de variáveis v_i, v_j define-se uma restrição binária $C_{i,j}$ que é um subconjunto de $D_i \times D_j$. Se $(u_{i,x}, u_{j,y}) \in C_{i,j}$ então a assertiva $\{v_i u_{i,x}, v_j u_{j,y}\}$ é consistente. Uma solução é uma assertiva $\{v_1 u_{i,w}, .., v_i u_{i,x}, .., v_j u_{j,y}, .., v_n v_n u_{n,z}\}$, tal que $\forall i, j : \{v_i u_{i,x}, v_j u_{j,y}\}$ é consistente. Cada predicado de sobreposição para a junção corresponde a uma restrição binária. Uma assertiva $\{v_1 u_{1,x}, v_2 u_{2,y}, v_3 u_{3,z}\}$ constitui uma solução se a cidade $u_{1,x}$ é cortada pela estrada $u_{2,y}$ que também corta a área de preservação $u_{3,z}$. As restrições são definidas através de um grafo Q, onde $Q[i][j]$ indica a condição de junção entre as relações R_i e R_j.

Uma forma de resolver este problema *CSP* seria fixar um valor para a variável cidade, depois atribuir valor para a variável estrada até achar uma estrada que cruza a cidade. Depois de fixar a cidade e a estrada verificar todas as áreas ecológicas (parte de avanço do algoritmo). Depois de esgotar todas as áreas de preservação o algoritmo retorna para as estradas e tenta encontrar dentre as estradas que sobraram uma que cruza a cidade fixada (fase de retorno do algoritmo). Existem técnicas para diminuir o número excessivo de verificações realizadas por este algoritmo no caso específico do mapeamento da junção

espacial M. Essas técnicas baseiam-se nas propriedades da indexação por árvores-R das relações da junção espacial e são a redução por janela *(window reduction WR)* e a busca sincronizada *(sincronous transversal ST)*.
A *WR* realiza uma busca sistemática utilizando janelas de consulta para encontrar os valores consistentes para as variáveis que não tenham sido instanciadas. Se for fixada uma cidade, esta cidade pode servir como janela de consulta para as estradas evitando checagens desnecessárias. A fase de avanço *WR* trabalha em uma forma de um loop aninhado indexado enquanto a fase de retorno pode ser baseada em vários algoritmos *CSP*.
A *ST* é uma generalização da junção baseada em árvore-R para um número arbitrário de entradas. O *ST* começa da raiz das árvores e tenta buscar combinações de entradas que satisfazem as restrições. Quando uma combinação válida é encontrada nos níveis intermediários, o algoritmo é chamado recursivamente tomando como parâmetro as referência dos nós intermediários, até que seja atingido o nível das folhas.

2.12.2 Modelo baseado na restrição do espaço de busca, na varredura de planos e nos predicados indiretos

O trabalho [31] propõe para a fase de filtragem um método de junção para várias relações denominado *Multi-way R-tree Join*. O *M-way R-tree Join* ($M > 2$) combina M relações espaciais através de $M - 1$ ou mais predicados espaciais percorrendo sincronamente M árvores R. Todas as relações devem estar indexadas por uma estrutura de árvore da família R. A ordem de realização da junção afeta o desempenho, por isso, como uma generalização das técnicas de restrição do espaço de busca e varredura de planos, são aplicadas as técnicas de ordem de restrição do espaço de busca *(SRO)* e ordem da varredura de planos *(PSO)*. Propõe-se também a técnica de filtragem de predicados indiretos *(IPF)*.

Ordem da restrição do espaço de busca

Sejam as relações diretas *REL1 intercepta REL2* e *REL2 intercepta REL3* e *REL3 intercepta REL4*. Na figura abaixo 2.9 como a relação *REL2* não tem interseção simultânea com as entradas nas relações *REL1* e *REL3* todas as relações serão descartadas quando aplica-se a restrição do espaço de busca.
A restrição do espaço de busca quando aplicada a várias relações segue uma estratégia para a ordem de escolha do nó que servirá de referência para a restrição *(space restriction ordering SRO)*. O procedimento *chooseNextNode*, descrito a seguir, do algoritmo *Space-Restriction* implementa esta escolha. Dado um conjunto de nós que possuem um predicado espacial entre si, é selecionado dentro deste conjunto, o nó com menor área comum de interseção com os outros nós e que não tenha ainda sido marcado. Se a área comum

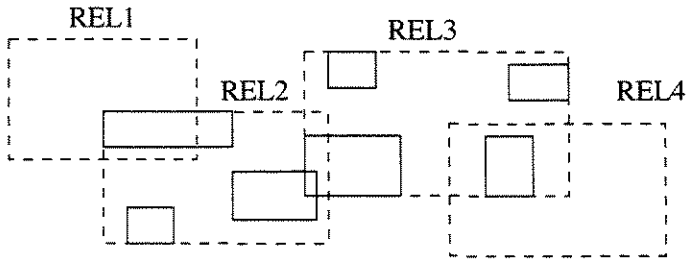


Figura 2.9: Interseção entre nós intermediários de árvores R

de interseção é zero para mais de um nó, destes nós seleciona-se o que tiver distância inter-retângulos máxima entre um par de outros nós que tem um predicado espacial com ele. Após a varredura marca-se este nó como visitado e o processo descrito continua até que todos os nós sejam marcados.

```
BOOLEAN SpaceRestriction(Query_graph q[][], RTree_nodes N[])
01 WHILE(TRUE) DO
02   i = chooseNextNode();        /* 0 ≤ i ≤ n−1 */
03   IF i == NULL THEN RETURN TRUE;  /* não há mais variáveis */
04   ReadPage(N[i]);
05   FOR all N_k ∈ N[i] DO
06     FOR j=0 TO n-1, i ≠ j DO
07       IF q[i][j]≠0 AND N_k.rect ∩ N[j].rect == ∅ THEN
08         N[i] = N[i] - N_k;
09         BREAK;
10   IF N[i]== ∅ THEN RETURN FALSE;
```

Ordem da varredura de planos

Na junção M, $M > 2$, o algoritmo de varredura dos planos é aplicado múltiplas vezes pois existem M variáveis e o número de predicados é maior ou igual a $M - 1$. Considere o grau como o número de predicados espaciais associadas a um nó. São definidos critérios para determinar a ordem de execução da varredura. Inicialmente todos os nós são inicializados como nós externos, escolhe-se os dois primeiros nós que tiverem um predicado espacial entre eles e cuja soma das cardinalidades seja mínima e realiza-se a varredura de planos entre eles marcado-os como nós internos. Escolhe-se um nó externo que satisfaça o predicado espacial com um ou mais nós internos e que possua um valor mínimo para a relação cardinalidade/ grau (quanto maior o número de predicados menor o resultado intermediário). Realiza-se a varredura entre este nó externo e o interno. Se existirem predicados adicionais entre esse nó externo e outros nós internos é feita à verificação destes

predicados. Este nó externo torna-se um nó interno. Realizam-se estas operações até que não existam mais nós externos.

Predicados indiretos

Considere as relações diretas *X intercepta Y e Y intercepta Z e Z intercepta W*. Podemos inferir a partir das relações diretas dadas e das propriedades dos *MBR's* associados aos objetos espaciais destas relações alguns predicados indiretos que podem ajudar no processo de poda e filtragem de soluções, esses predicados indiretos referem-se as relações implícitas entre X e Z ou Y e W, por exemplo. Considere b_{jx} como o valor máximo de extensão para o eixo x dos objetos de Z e que $a \in X\, e, b \in Z$, temos a seguinte condição indireta $x_{dist}(a, c) \leq b_x$, se esta condição não for satisfeita para um par de nós conclui-se que este par de nós não faz parte da solução. Estas condições indiretas podem ser produzidas a partir de um grafo de conexão entre as diversas relações onde cada relação corresponde a um vértice e existem arestas entre os vértices se existirem predicados espaciais entre as relações. O peso de uma aresta equivale ao valor máximo de distância entre dois objetos das relações que estão nos vértices. Pode-se utilizar o menor caminho no grafo entre dois vértices para inferir, baseando-se nos pesos das arestas do caminho, relações indiretas entre as relações associadas aos vértices dos extremos do caminho. A poda utilizando os predicados indiretos denomina-se *indirect predicate filtering (IPF).*

Capítulo 3

Survey sobre métodos analíticos para junção-R

3.1 Modelo de Faloutsos, Sellis e Roussopoulos

Este é o primeiro trabalho [11] a propor um método analítico para prever o desempenho do método de indexação baseada em árvores da família R, mais especificamente baseados em árvores R e R+. As fórmulas são derivadas a partir de um modelo de dados composto de segmentos de reta de tamanho fixo, com distribuição uniforme, definidos em um espaço unidimensional [0,1]. Um segmento de reta de tamanho q é usado como uma janela de consulta, procura-se estimar o número de segmentos interceptados pelo segmento de tamanho q, bem como o número de acessos a nó necessário para verificar esta interseção considerando os dados indexados por uma estrutura baseada nas árvores R e R+.

Para facilitar a análise aplicou-se uma transformação dos objetos para um espaço de maior dimensionalidade. Os segmentos indicados por $x_{inicial}$ e x_{final} são transformados em pontos da forma ($x_{inicial}$, x_{final}) para um espaço bidimensional. Assume-se também que todos nós da árvore estão cheios de dados *(packed trees)*, a técnica de *packing* utilizada é aplicada a banco de dados relativamente estáticos e serve para diminuir o *overlap* e *dead space* na estrutura de indexação.

Tanto os resultados obtidos pelos métodos analíticos propostos quanto os dados experimentais levantados mostraram um melhor desempenho da árvore-R+ sobre a árvore-R. Para a análise considerou-se *coverage* como a área dos *MBR's* de todos os nós folhas da árvore-R e *overlap* a área total contida dentro de dois ou mais nós folhas da árvore-R.

A análise pode ser simplificadas através da aplicação de transformações, por exemplo, paralelepípedos são levados para pontos no espaço quadrimensional. Para paralelepípedos alinhadas com os eixos, 4 coordenadas são suficientes para determina-los unicamente, as coordenadas y e x do canto inferior esquerdo e superior direito. Os retângulos são

transformados em objetos unidimensionais representados por segmentos de reta, esses segmentos são transformados em pontos no espaço bidimensional.

A análise baseou-se também nas seguintes premissas:

- Os segmentos de reta de um dado tamanho estão uniformemente distribuídos e não precisam estar contidos completamente no intervalo .
- O ponto de inicio destes segmentos dividem o intervalo (-σ,1) em N+1 subintervalos iguais , onde N é o número de segmentos e σ o seu tamanho. O objetivo desta distribuição foi manter uma sobreposição *(overlap)* constante no intervalo.

A análise abordou dois casos: *one-size* e *two-size*. No *one-size* foram considerados N segmentos de tamanho σ uniformemente distribuídos. No *two-size* foram considerados dois conjuntos de segmentos uniformemente distribuídos, sendo o primeiro conjunto com N_1 segmento de tamanho σ_1 e o segundo conjunto com N_2 segmentos de tamanho σ_2. Segue abaixo a análise proposta:

Lema : Dados N segmentos de tamanho σ , uniformemente distribuídos no intervalo, um segmento de reta *(query segment)* de tamanho q intercepta :

$$intersect(N, \sigma, q) = \frac{(\sigma + q)}{(1 + \sigma)}(N + 1) \tag{3.1}$$

Colorário: Dados N segmentos de tamanho σ , uniformemente distribuídos no intervalo, o *overlap* é constante e igual a:

$$O_v(N, \sigma) = \frac{\sigma}{(1 + \sigma)}(N + 1) \tag{3.2}$$

3.1.1 Análise da árvore-R+

Sejam N segmentos de tamanho σ, distribuídos uniformemente. Sendo h_+ a altura da árvore-R+ que está cheia, isto é, cada página de dados contém C entradas e cada nó interno possuí f filhos. O número total de páginas de dados é f^{h_+}, dividindo o espaço em f^{h_+} intervalos. Devido à distribuição uniforme, cada intervalo terá um tamanho $\frac{1}{f^{h_+}}$. Cada página corresponde a um desses intervalos. Os segmentos que interceptam um dado intervalo serão inseridos na respectiva página.

$$C = intersect(N, \sigma, \frac{1}{f^{h_+}}) \tag{3.3}$$

ou

$$C = O + \frac{1}{f^{h_+}}(N + 1 - O) \tag{3.4}$$

ou

$$h_+ = \log_f \frac{N + 1 - O}{C - O} \tag{3.5}$$

Onde O é o *overlap* $O = O_v(N, \sigma)$ do conjunto de segmentos. O número total de acessos a disco r^+_{da} para consultar quantos segmentos contém um dado ponto é igual à altura da árvore mais um , para contar a busca da página de dados. Assume-se que a raiz da árvore está no disco.

$$r^+da = 1 + \log_f \frac{N + 1 - O}{C - O} \tag{3.6}$$

, ou tendo $N \gg 1$ e $N \gg 0$

$$r^+da \approx 1 + \log_f \frac{N}{C - O} \tag{3.7}$$

3.1.2 Análise para árvore-R

Considerando a árvore cheia temos os segmentos agrupados em $\frac{N}{C}$ páginas, em grupos de tamanho C. Cada página é caracterizada pelo segmento que cobre todos os segmentos na página. Estes segmentos envolventes são transformados em pontos no espaço. Demonstra-se que esses segmentos serão iguais em tamanho $\sigma_{pai,1}$ e tem distribuição uniforme.

$$\sigma_{pai,1} = \frac{1 + \sigma}{N + 1}(C - 1) + \sigma \tag{3.8}$$

As páginas de dados são agrupadas em $\frac{N}{(Cf)}$ grupos , cada grupo contendo f páginas de dados. Os pais no nível i tem tamanho $\sigma_{pai,i}$ e tem distribuição uniforme. A raiz da árvore corresponde ao pai do nível *h+1*.

$$\sigma_{pai,i} = \frac{1 + \sigma}{N + 1}(Cf^{i-1} - 1) + \sigma \tag{3.9}$$

3.2 Modelo baseado em árvores empacotadas

Este trabalho [20] tem como objetivo otimizar a utilização de espaço para uma árvore-R resultando numa redução do espaço nos nós da árvore. Esta redução produz uma melhora no tempo de resposta para uma consulta a um objeto indexado pela árvore, já que a árvore terá um maior *fan-out* e poderá ter sua altura diminuída. Considerando um conjunto estático de dados, o empacotamento da árvore é implementado através da reconstrução de baixo para cima da mesma, isto é, os retângulos dos nós folha são ordenados segundo uma *space filling curve* e depois são reagrupados a partir desta ordenação para a reconstrução da árvore de acordo com o novo reagrupamento. Uma *space filling curve* é uma curva que visita todos os pontos de um *grid* k-dimensional exatamente uma vez e nunca cruza com ela mesma. Este mesmo trabalho [20] foi constatado a a aplicação da curva de *Hilbert*.

O trabalho também propõe um modelo analítico para prever o tempo médio de resposta para uma consulta de janela *(range query)* sobre um conjunto de dados indexados por uma estrutura baseada na árvore-R. As fórmulas propostas dependem da área e perímetro dos nós da árvore-R.

3.2.1 Modelo analítico

As fórmulas propostas são válidas para todas as variantes das árvores R e são independentes dos detalhes de implementação e manutenção da árvore. Assume-se que consultas correspondem a áreas retangulares uniformemente distribuídas no espaço. O tempo de resposta medido para uma consulta é baseado no número de acessos a nós da árvore.

Lema 1. Seja $n_{i,x} \times n_{i,y}$ o *MBR* para o nó n_i de uma árvore-R, temos que a probabilidade deste nó ser acessado em uma consulta a ponto *(point query)* é $DA(n_{i,x}, n_{i,y})$.

$$DA(n_{i,x}, n_{i,y}) = n_{i,x} \times n_{i,y} \tag{3.10}$$

Prova (Lema 1). Tendo-se um distribuição uniforme das janelas de consulta e considerando $WS = [0,1)^d$ a unidade dimensional do espaço com área 1, a probabilidade de uma janela de consulta conter um ponto do espaço é igual à área da janela dividida pela área total do espaço, isto é, $\frac{n_{i,x} \times n_{i,y}}{1}$.

Lema 2. O número esperado de acessos a disco $P(0,0)$ para uma consulta a ponto é:

$$P(0,0) = \sum_{i=1}^{N} n_{i,x} \times n_{i,y} \tag{3.11}$$

Prova (Lema 2). Por 3.10 cada nó da árvore-R contribui com $DA()$ acessos a disco, temos portanto que somar $DA()$ para todos os nós da árvore.

Lema 3. O número esperado de acessos a disco $P(q_x, q_y)$ para uma consulta a uma janela retangular $q_x \times q_y$ é:

$$P(q_x, q_y) = \sum_{i=1}^{N} n_{i,x} \times n_{i,y} + q_y \times \sum_{i=1}^{N} n_{i,x} + q_x \times \sum_{i=1}^{N} n_{i,y} + N \times q_x \times q_y \quad (3.12)$$

Se considerarmos $q_x = q_y = 0$ a equação 3.12 se reduz à equação 3.11 de consulta a ponto. Considerando a soma das extensões de todos os nós da árvore-R na direção do eixo x igual a L_x e no eixo y igual a L_y e $AreaTotal$ a área total dos nós da árvore-R temos:

$$P(q_x, q_y) = AreaTotal + q_x \times L_y + q_y \times L_x + N \times q_x \times q_y \quad (3.13)$$

Esta equação 3.13 evidencia a importância de se minimizar o perímetro e a área dos nós da árvore-R.

Prova (Lema 3). Uma consulta retangular de tamanho $q_x \times q_y$ é equivalente a uma consulta a ponto, se inflarmos os nós da árvore-R por q_x na direção do eixo x e q_y na direção do eixo y. Assim, o nó n_i com tamanho $n_{i,x} \times n_{i,y}$ comporta-se como um nó de tamanho $(n_{i,x} + q_x) \times (n_{i,x} + q_y)$. Logo:

$$P(q_x, q_y) = \sum_{i=1}^{n} (n_{i,x} + q_x) \times (n_{i,x} + q_y) \quad (3.14)$$

3.3 Modelo analítico para consultas a janela

O trabalho de Pagel [28] propõe um método analítico para prever o número de acessos a *bucket* necessários para responder uma consulta de janela *(range query)*. O modelo analítico proposto é independente da estrutura dos dados e dos detalhes de implementação da estrutura de indexação. Foram caracterizados dois tipos possíveis de janela e dois tipos de comportamento de usuário de consulta, a combinação desses tipos gera quatro tipos diferentes de ambiente que servem como base para a análise. Os tipos de janela compõe-se de um primeiro tipo onde cada parte da janela pode ser requisitada com igual probabilidade e um outro no qual cada objeto tem a mesma probabilidade de ser acessado. Quanto ao tipo de usuário foram caracterizados o usuário que especifica o valor da janela em termos da área total e o usuário que sempre determina o valor de retorno da consulta com a intenção de trazer sempre um mesmo número de objetos.

Considera-se que os *MBRs* dos objetos espaciais estão agrupados nos *buckets* obedecendo a uma relação de proximidade espacial dos *MBRs* dentro de cada *bucket*. Os dados compões-se tanto de pontos quanto de retângulos. A consulta analisada é uma *range query* que busca os pontos que estão na janela de consulta e os retângulos que tem interseção com a janela de consulta.

3.3.1 Modelo analítico

Define-se um modelo de janela de consulta *(window query model WQM)* da forma:

$WQM = (ar, M, c_m, F_c)$, onde ar é a razão entre as proporções da janela de consulta, a qual foi considerada $1 : 1$ para todos os modelos propostos, isto é, as janelas são quadradas, M corresponde ao tamanho da janela, c_m é um valor constante que indica o valor da janela e F_c é uma função que caracteriza a distribuição dos centros das janelas de consulta.

Considerando k cada um dos tipos de janela de consulta WQM_k $(1 \leq k \leq 4)$, para o conjunto de buckets $B = \{B_1, .., B_m\}$ e $R(B) = \{R(B_1, ..R(B_m)\}$ a organização do espaço de dados, define-se $P_k(w \cap R(B); j)$ a probabilidade de uma janela w interceptar exatamente j *bucktes* em $R(B)$. No modelo proposto, o número esperado de acessos a *bucket* que interceptam uma janela de consulta chamado *PM (performance measure)* é:

$$PM(WQM_k, R(B)) = \sum_{j=0}^{m} j \times P_k(w \cap R(B); j) \tag{3.15}$$

Sendo $P_k(w \cap R(B_i) \neq 0)$ a probabilidade da janela w interceptar a região de *bucket* $R(B_i)$, prova-se [28] que:

$$\sum_{j=0}^{m} j \times P_k(w \cap R(B); j) = \sum_{i=1}^{m} P_k(w \cap R(B_i) \neq 0) \tag{3.16}$$

3.4 Consultas a ponto e dimensão fractal

Este trabalho [10] propõe um modelo para análise de desempenho de estruturas de indexação baseadas na árvore-R. O modelo foi proposto para dados compostos de pontos no espaço, sendo os dados indexados por uma estrutura baseada na árvore-R. Considerando que nas aplicações espaciais reais os dados apresentam um distribuição assimétrica, as fórmulas propostas aplicam-se a pontos com distribuição assimétrica no espaço. Para equacionar a distribuição assimétrica de pontos no espaço foi usado o conceito de fractal. Utilizando a dimensão fractal de um conjunto de pontos e o número de pontos do conjunto derivou-se a fórmula que estima o número de acessos a disco para consultas a janela *(range queries)*.

3.4.1 Modelo analítico

Mostrou-se por amostragem, que dados espaciais de um aplicação real apresentam um comportamento fractal, ou seja, eles apresentam a propriedade da auto-similaridade. A propriedade de auto-similaridade para um conjunto de pontos caracteriza-se quando

porções do conjunto de pontos são estatisticamente iguais ao conjunto completo.
Seja um objeto geométrico que possui a propriedade de auto-similaridade, consistindo num conjunto de pontos num espaço D-dimensional. Divide-se o espaço D-dimensional em células de grade (hiper-)cúbicas de tamanho r. Sendo $N(r)$ o número de pontos que estão dentro de cada célula, isto é, o número de células penetradas pelo fractal, temos:
Definição. A dimensão fractal d para um conjunto de pontos que apresenta a propriedade de auto similaridade é:

$$d = \frac{\partial \log(N(r))}{\partial \log(r)} = constante \tag{3.17}$$

Colorario. Para um conjunto de pontos com a propriedade de auto-similaridade e constante de integração K, o número de células penetradas pelo fractal $N(r)$ é:

$$N(r) = \frac{K}{r^d} \tag{3.18}$$

Assumindo que o endereçamento do espaço é normalizado para o hiper-cubo unitário, temos $K = 1$:

$$N(r) = \frac{1}{r^d} \tag{3.19}$$

A capacidade efetiva dos nós da árvore-R C_{eff} é igual ao número médio de entradas por nó. Sendo u a média de utilização dos nós e C sua capacidade total, temos:

$$C_{eff} = C \times u \tag{3.20}$$

Sendo $s = (s_1, .., s_D)$ o tamanho esperado dos MBRs dos nós folha. Assumindo que a árvore-R gera *MBRs* quadrados, temos, $s_1 = .. = s_D \equiv \sigma$. A partir de 3.18 chagamos a:

$$N_l = \frac{1}{\sigma^d} \tag{3.21}$$

Temos também que o número de nós folhas é igual ao número total de objetos dividido pela capacidade media dos nós:

$$N_l = \frac{N}{C_{Eff}} \tag{3.22}$$

$$\frac{N}{C_{eff}} = \frac{1}{\sigma^d} \tag{3.23}$$

$$\sigma = (\frac{C_{eff}}{N})^{\frac{1}{d}} \tag{3.24}$$

Como visto em [20], o número médio de nós acessados por uma janela de consulta de tamanho q:

$$P(q) = \sum_{n} \prod_{i=1}^{D} (x_{i,n} + q_i) \tag{3.25}$$

$$P(q) = \sum_{todos\ nos\ folhas} \prod_{i=1}^{D} (\sigma + q_i) = \frac{N}{C_{eff}} \prod_{i=1}^{D} (\sigma + q_i) \tag{3.26}$$

Considerando uma árvore-R de altura h onde a raiz da árvore está no nível $j = 0$ e as folhas estão no nível $j = h - 1$. O número total de acessos a nós é dado pela adição dos nós acessados a cada um dos níveis da árvore.

$$P_{all}(q) = \sum_{j=0}^{h-1} \frac{N}{C_{eff}^{h-j}} \prod_{i=1}^{D} (\sigma_j + q_i) \tag{3.27}$$

$$\sigma_j = (\frac{C_{eff}^{h-j}}{N})^{\frac{1}{d}}, j = 0, ..., h-1 \tag{3.28}$$

A dimensão fractal está baseada em uma teoria já consolidada e usa apenas um número para descrever um conjunto não uniforme de pontos, sendo a dimensão fractal da distribuição uniforme caracterizada por $(d = D)$.

3.5 Limite inferior para consultas a janela

Este trabalho [29] busca um limite inferior para o desempenho de um consulta de janela a um conjunto de dados espaciais indexados por um estrutura baseada na árvore-R. Este limite inferior serve como um ponto de comparação absoluta entre os outros métodos já existentes. Para simplificar, o método foi implementado utilizando dados estáticos, pois um limite inferior para dados estáticos também é um limite inferior para os dinâmicos.

3.5.1 Modelo

Definiu-se a medida para comparação de desempenho para estruturas de indexação como sendo o número de acessos a *buckets (PM)* necessário para realizar uma consulta de janela

(range query) onde os dados estão armazenados em *buckets*. Esta medida foi escolhida por ser independente da estrutura de dados empregada e dos detalhes de implementação. Busca-se uma otimização para o agrupamento de dados espaciais, que servirá como base para uma comparação absoluta de desempenho para as estruturas já conhecidas.
A região do *bucket* é o menor intervalo d-dimensional que engloba todos os objetos armazenados no *bucket*. Assumindo que para armazenar um conjunto $G = (g_1, .., g_n)$ de n objetos a estrutura de dados $DS(G)$ utiliza m *buckets* de dados $B_1, .., Bm$ com capacidade de c_b objetos cada. Utilizando um modelo de janela QM [28], o problema pode ser assim definido:
Dado um conjunto de objetos espaciais, uma capacidade de *bucket* $C_b \leq 2$, e um modelo de janela QM, determinar o conjunto de *buckets* B_{opt} para o qual o número de acessos a *bucket* de uma consulta PM é mínimo.
Para buckets com tamanho máximo igual a dois demonstra-se que o problema é polinomial. Prova-se que o problema é NP-difícil quando $C_b \leq 2$, por isso emprega-se um heurística como uma melhor solução factível para o problema. A heurística aplicada foi à simulação de anel *(simulated annealing heuristics)* por se tratar de um algoritmo geral, apropriado para todos os tipos de consulta e que possui uma implementação simples. Os resultados apresentados pelo modelo baseado na heurística diferenciam-se em 5% do modelo ótimo, este é um valor típico para casos de aplicação desta heurística. Para as implementações conhecidas para dados estáticos, a árvore LSD obteve 10% de diferença com a solução ótima, já a árvore-R empacotada *(packed R-tree)* apresentou 30% de diferença.

3.6 Fator *Buffer* nas consultas a janela

Este trabalho [21] inova por incluir o fator *buffer* na análise de desempenho das consultas espaciais. Os autores propõem como medida de desempenho para consulta o número de acessos a disco ao invés do número de acesso a nó, geralmente 1 nó está armazenado em uma página de disco. São analisadas consultas baseadas em janelas retangulares uniformemente distribuídas no espaço. Baseando-se nesse modelo propõe-se uma correção para as fórmulas analíticas propostas em trabalhos anteriores [20] e também novas fórmulas de desempenho que levam em conta o fator *buffer*.

3.6.1 Modelo proposto

O número de acessos a nó não é uma boa medida para o desempenho de uma consulta por janela a dados indexados por uma estrutura baseada em árvores R. O tempo de acesso à memória é muito menor que o tempo de acesso a disco, por isso, se um nó estiver na memória *(buffer)* o acesso a ele não é um fator determinante do desempenho da consulta.

Considera-se que o desempenho pode ser corretamente avaliado pelo número de acessos a disco.

No modelo, cada página de disco tem capacidade de armazenar um nó da árvore de indexação, portanto um acesso a disco corresponde a um acesso a um nó. Existe um espaço restrito reservado na memória *(buffer)*, a utilização dos nós armazenados no *buffer* diminui o tempo de consulta em comparação às consultas a nó em disco.

No modelo matemático proposto em [20], considerando o espaço unitário e dada uma janela de consulta e um retângulo dentro deste espaço, formulou-se que a probabilidade de interseção entre a janela de consulta e o retângulo é igual à probabilidade do canto superior direto da janela estar neste mesmo retângulo quando os lados do retângulo são expandidos pelos comprimentos da janela de consulta em cada direção.

Para janelas de consulta retangulares uniformemente distribuídas no espaço de tamanho fixo $q_x \times q_y$ o canto superior direito Q_{tr} da região de consulta Q não pode ser um ponto qualquer dentro do espaço, pois a janela de consulta deve estar completamente dentro do espaço total, isto é, Q_{tr} deve estar dentro da área $U' = [q_x, 1] \times [q_y, 1]$. Por isso, a probabilidade de acessar um retângulo $R = (a, b), (c, d)$ é igual à porcentagem de U' coberta pelo retângulo $R' \cap U'$:

$$A_{i,j}^{Q} = \frac{area(R' \cap U')}{area(U')} = \frac{[\min(1, c + q_x) - \max(a, q_x)][\min(1, d + q_y) - \max(b, q_y)]}{(1 - q_x)(1 - q_y)} \quad (3.29)$$

A partir da equação acima foi feito o modelamento baseado também no fator *buffer*, isto é, foi calculado apenas os acessos a nós no disco retirando-se do calculo os acessos ao *buffer*. As páginas do *buffer* são gerenciadas pela política *LRU*. Considerou-se que a probabilidade de encontrar uma página no *buffer* estabilizado é igual à probabilidade de se encontrar uma página no *buffer* quando o *buffer* fica cheio pela primeira vez.
Considerando que a probabilidade de acessar um retângulo $R_{i,j}$ enquanto executa-se uma consulta é $A_{i,j}$. Sendo assim a probabilidade de não acessar o retângulo $R_{i,j}$ para N consultas é $P[B_{i,j} = 0|N] = (1 - A_{i,j})^N$. Então a probabilidade de $R_{i,j}$ ser acessado pelo menos uma vez nas próximas N consultas é $P[B_{i,j} \geq 1|N] = 1 - (1 - A_{i,j})^N$ e o número esperado de acessos a disco em N consultas é:

$$D(N) = \sum_{i=0}^{H} \sum_{j=1}^{M_i} P[B_{I,j} \geq 1|N] = M - \sum_{i=0}^{H} \sum_{j=1}^{M_i} (1 - A_{i,j})^N \quad (3.30)$$

Sendo que $(D(0) = 0 < B)$ e $(D(1) =$ a soma da área de todos *MBRs* da árvore). Seja N^* o número de consultas necessárias para encher o *buffer* e deixa-lo na situação estável. Depois do *buffer* ficar estabilizado a probabilidade da consulta acessar um retângulo no

disco é igual à probabilidade da consulta acessar um nó e esse não estar no *buffer*:

$$\sum_{i=0}^{H}\sum_{j=1}^{M_i} A_{i,j} \times P[B_{i,j} = 0|N^*] = \sum_{i=0}^{H}\sum_{j=1}^{M_i} A_{i,j} \times (1 - A_{i,j})^{N^*} \tag{3.31}$$

Temos que o número esperado de acessos a disco necessário para responder uma consulta à janela de tamanho $q_x \times q_y$ é:

$$E_{D,T}(q_x, q_y) = \sum_{i=0}^{H}\sum_{j=1}^{M_i} A^Q_{i,j} \times (1 - A^Q_{i,j})^{N^*} \tag{3.32}$$

3.7 O modelo de Huang e Jing para desempenho de junção espacial

O trabalho de Huang [15] é o primeiro a propor um método analítico para análise de desempenho de junções espaciais. O modelo baseia-se na busca em profundidade. O modelo matemático estima primeiramente o número de acessos desconsiderando a existência de um *buffer*. Uma função de distribuição de probabilidade é utilizada para estimar o número de acessos ao *buffer*.

3.7.1 Modelo

Sejam duas árvores R e $\tilde{R}$ de mesma altura h utilizando *MBRs* para indexar dois conjuntos de dados e uma busca em largura que percorre esses índices para verificar a junção. Se considerarmos que não existe *buffer*, isto é, o tamanho do *buffer* $b = 0$, temos que, baseando-se nos modelos descritos nas seções anteriores, o número de acessos a disco necessário para realizar a junção é:

$$ZEIO_h = 2 + 2 \times \sum_{l=1}^{h-1}\sum_{i=1}^{N^l}\sum_{j=1}^{\tilde{N}^l}(x_i^l + \tilde{x}_i^l) \times (y_i^l + \tilde{y}_j^l) \tag{3.33}$$

O primeiro fator constante 2 da equação refere-se a busca da raiz das duas árvores na memória. Depois para cada nível l da árvore, para cada nó i de um dado nível e para todas as combinações possíveis para aquele nó com os outras nós j da outra árvore é feita à verificação de interseção. Cada vez que se faz a busca dois nós são trazidos da memória, um para cada árvore, por isso o fator 2 multiplica a equação.
Se considerarmos que as árvores tem tamanho diferente h e $\tilde{h}$ sendo $h < \tilde{h}$, temos apenas que fazer a busca como era feita anteriormente até a árvore menor atingir os nós folhas e

depois usamos esses nós folhas como janela de consulta para a busca nos níveis restantes da árvore mais alta:

$$ZEIO_{(h<\tilde{h})} = ZEIO_h + 2 \times \sum_{l=h}^{\tilde{h}-1} \sum_{i=1}^{N^h} \sum_{j=1}^{\tilde{N}^l} (x_i^{h-1} + \tilde{x}_i^{\tilde{l}}) \times (y_i^{h-1} + \tilde{y}_j^{l}) \tag{3.34}$$

Modelo probabilístico de acesso ao *buffer*

Da equação acima proposta para acesso a nó da árvore, para efeitos de análise de desempenho devem ser retirados os acessos que encontraram o nó no *buffer*. Considerando um *buffer* de tamanho b gerenciado pela política LRU da manutenção de páginas, ao se buscar um nó no *buffer* este nó não estará no *buffer* se for a primeira vez que ele é acessado, ou se desde a última vez que ele foi acessada já foram acessado b ou mais nós diferentes.
Definição. dado um nó N, então o número esperado de vezes em que ele não é encontrado no *buffer (page fauls)* $E(N)$ é:

$$E(N) = \begin{cases} 1, & se\ for\ o\ primeiro\ acesso \\ P\{(x \geq b)\}, & nos\ outros\ casos \end{cases} \tag{3.35}$$

Considere que o número de acessos consecutivos sem sucesso ao *buffer* em um intervalo é pequeno. Seja *Inter-access Page Fault* o número de buscas sem sucesso ao *buffer* entre dois acessos consecutivos a N. Considerou-se o IPF pequeno baseando-se no fato da árvore-R preservar uma alta localidade espacial e porque os percursos na árvore geralmente utilizam otimizações. Foi atribuído a IPF uma variável de valor randômico com distribuição exponencial. A distribuição $f(x) = \lambda e^{-\lambda x}$ onde $x \geq 0$ e $\lambda > 0$ tem função de distribuição cumulativa $F(x) = 1 - e^{-\lambda x}$, para $x \geq 0$. Sendo $\lambda = \frac{1}{\mu}$ onde μ indica o IPF. O valor para μ foi definido experimentalmente.
Se considerarmos agora o *buffer* de tamanho b, a equação final para acesso a disco para junção é igual ao número de acessos a nó quando os nós são acessados pela primeira vez, portanto estão no disco, mais o número de acessos aos nós já acessados pelo menos uma vez sendo que este nós não estão no *buffer*.

$$EIO = n + m + (ZEIO - n - m) \times Px \geq b \tag{3.36}$$

O fator n+m da equação corresponde a primeira busca de um nó, que neste caso nunca está no *buffer*. Foi considerado, para efeitos de análise, que de um modo geral uma junção acessa todos os nós das árvores. A comparação entre resultados de implementações e o método analítico proposto chegaram a um valor de 6% de erro quando o *buffer* possuía um tamanho pequeno e a 20% para *buffers* de maior tamanho.

(3.37)

3.8 O modelo de Theodoridis, Stefanakis e Sellis para junção espacial

Estes trabalhos [36, 35, 37] apresentam um modelo analítico para estimar o custo em termos de acessos a nó e a disco para consultas seleção e junção, usando estruturas de indexação baseadas em árvores R . As fórmulas propostas não dependem da estrutura da árvore-R e são aplicadas a distribuições uniformes e não-uniformes de dados sendo baseadas no algoritmo de busca em profundidade.

3.8.1 Consultas de seleção *(selection queries)*

Formalmente, o problema da análise de custo da árvore-R para consultas de seleção é definido da seguinte maneira: Sendo d a dimensão do espaço de dados e $WS = [0,1)^d$ a unidade dimensional do espaço. Seja R uma árvore de altura h, onde a raiz está no nível h_R e as folhas no nível 1,com o nó raiz armazenado na memória. Considerando N_R retângulos armazenados em um índice da árvore R, busca-se os retângulos deste conjunto que sobrepõe *(overlap)* uma janela de consulta $q = (q_1, ..., q_d)$.

Definição. A densidade D de um conjunto de N retângulos com tamanho médio $s = (s_1, ..., s_d)$ é o número médio de retângulos que contém um dado ponto no espaço d-dimensional. Essa densidade pode ser expressa como a soma das áreas de todos os retângulos dividida pela área total do espaço. Como temos o espaço $WS = [0,1)^d$ de área igual a 1:

$$D(N, s) = \sum_{N} \prod_{k=1}^{d} s_k = N. \prod_{k=1}^{d} s_k \tag{3.38}$$

Seja $N_{R,l}$ o número de nós no nível l e $s_{R,l}$ o tamanho médio dos nós no nível l. Como o número de acessos a nós é igual o número de nós com interseção, o número esperado de acessos a nó $NA\_total(R, q)$ é:

$$NA\_total(R, q) = \sum_{l=1}^{h_R - 1} intersec(N_{R,l}, s_{R,l}, q), \tag{3.39}$$

onde $intersec(N_{R,l}, s_{R,l,q})$ é a função que retorna o número de nós no nível l interceptados pela janela de consulta q.

Lema. Dado um conjunto de N retângulos $r_1, ..., r_N$ com tamanho médio s e um retângulo r com tamanho q, o número médio de retângulos interceptados por r é:

$$intersec(N, s, q) = N.\prod_{k=1}^{d}(s_k + q_k) \tag{3.40}$$

Prova. Dado um conjunto de N retângulos com tamanho médio s e um retângulo r com tamanho q, o número médio de retângulos deste conjunto que interceptam o retângulo r é igual ao número de retângulos de um segundo conjunto de N retângulos de tamanho médio s' que contém um ponto no espaço, onde $s'_K = s_k + q_k, \forall k$. Por definição, está é a densidade D' do segundo conjunto de retângulos.

$$intersec(N, s, q) = intersec(N, s', 0) = D'(N, s') = N.\prod_{k=1} ds'_k \tag{3.41}$$

Assumindo que o retângulo r representa um janela de consulta, a partir das equações 3.39 e 3.40 temos que $NA\_total(R, q)$:

$$NA\_total(R, q) = \sum_{l=1}^{h_R-1}\left\{N_{R,l}.\prod_{k=1}^{d}(s_{R,l,k} + q_k)\right\} \tag{3.42}$$

As propriedades h_R, $N_{R,l}$ e $s_{R,l,k}$ da árvore podem ser expressas em função das propriedade cardinalidade N_{R_1} e densidade do conjunto D_{R_1}. A altura h_R de uma árvore R com capacidade média dos nós *(fan-out f)* que armazena N_R retângulos é [11]:

$$h_R = 1 + \left\lceil \log_f \frac{N_R}{f} \right\rceil \tag{3.43}$$

Como um nó organiza, em média, f retângulos, o número médio de nós no nível l da árvore R é:

$$N_{R,l} = \frac{N_R}{f^l} \tag{3.44}$$

Considerando retângulos com lados de mesmo tamanho, isto é, quadrados ($s_{R,l,1} = ... = s_{R,l,d}, \forall l$) [20].

$$D_{R,l} = N_{R,l}.\prod_{k=1}^{d} s_{R,l,k} = \frac{N_R}{f^l}.(s_{R,l,k})^d \Rightarrow s_{R,l,k} = (D_{R,l}.\frac{f^l}{N_R})^{\frac{1}{d}} \tag{3.45}$$

Supondo que no nível l, $N_{R,l}$ nós com tamanho médio $(s_{R,l,k})^d$ estão organizados em $N_{R,l+1}$ nós pais com tamanho médio $(s_{R,l+1,k})^d$. Cada nó pai agrupa em média f nós filhos e $f^{\frac{1}{d}}$ são responsáveis pelo tamanho do nó pai para cada direção. Assumindo que os centros das projeções dos $N_{R,l}$ retângulos estão igualmente distanciados e essa distância $t_{l,k}$ depende dos $N_{R,l}^{\frac{1}{d}}$ nós para cada direção. O tamanho médio $s_{R,l+1,k}$ de um nó pai em cada direção é:

$$s_{R,l+1,k} = (f^{\frac{1}{d}} - 1).t_{l,k} + s_{R,l,k}, \tag{3.46}$$

onde $t_{l,k}$ representa a distância entre os centros das projeções dos retângulos consecutivos na dimensão k.

$$t_{l,k} = \frac{1}{N_{R,l}^{\frac{1}{d}}}, \forall l \tag{3.47}$$

Lema. Dado um conjunto de nós $N_{R,l}$ com densidade $D_{R,l}$, a densidade $D_{R,l,k}$ de suas projeções é:

$$D_{R,l,k} = (D_{R,l}.N_{R,l^{d-1}})^{\frac{1}{d}} \tag{3.48}$$

Prova:

$$D_{R,l,k} = N_{R,l} \times s_{R,l,k} \Rightarrow \prod_d (N_{R,l} \times s_{R,l,k}) = N_{R,l}^d \times \prod_d s_{R,l,k} \Rightarrow \tag{3.49}$$

$$\Rightarrow (D_{R,l,k})^d = N_{R,l}^d \times \frac{D_{R,l}}{N_{R,l}} = (N_{R,l})^{d-1} \times D_{R,l} \tag{3.50}$$

Portanto o número total de acessos pode ser derivado utilizando somente as propriedades dos dados quantidade N_R e densidade D_R, do *fan-out* dá árvore f e extensão q da janela de consulta:

$$NA\_total = \sum_{l=1}^{1+\lceil \log_f \frac{N_R}{f} \rceil} \left\{ \frac{N_R}{f^l} \times \prod_{k=1}^{d} ((D_{R,l} \times \frac{f^l}{n_R})^{\frac{1}{d}} + q_k) \right\} \tag{3.51}$$

3.8.2 Consultas de junção espacial

A análise de custo proposta fornece uma fórmula que estima o número médio de nós acessados para processar um consulta de junção entre dois conjuntos de dados, baseada

somente no conhecimento da propriedade dos dados, sem se preocupar com a estrutura da árvore R.
A análise baseia-se no algoritmo de junção espacial SJ abaixo descrito. SJ realiza uma busca sincronizada em ambas as árvores, sendo que as entradas dos nós R_1 e R_2 fazem os papeis de dados e retângulos de consulta. Para ambas as operações, o custo total é medido pela quantidade total de acessos a páginas no índice da árvore R, no algoritmo SJ um acesso corresponde a uma chamada a *ReadPage*.

```
SJ(R1,R2:nosArvoreR) /* Algoritmo de busca em profundidade */
01 BEGIN
02   FOR all Er2 in R2 DO
03     FOR all Er1 im R1 DO
04       IF Er1.rect overlaps Er2.rect THEN
05         IF R1 and R2 are leaf pages THEN
06           Output(Er1.oid,Er2.oid)
07         ELSE IF R1 is a leaf node THEN
08           ReadPage(Er2.ptr);
09           SJ(Er1.ptr,Er2.ptr)
10         ELSE IF R2 is a leaf node THEN
11           ReadPage(Er1.ptr);
12           SJ(Er1.ptr,Er2.ptr)
13         ELSE
14           ReadPage(Er1.ptr,Er2.ptr);
15           SJ(Er1.ptr,Er2.ptr)
16         END-IF
17       END-IF
18     END-FOR
19   END-FOR
20 END.
```

Formalmente, o problema da análise de custo das consultas de junção pode ser definido da seguinte maneira: Sendo d a dimensionalidade do espaço e $WS = [0,1)^d$ a unidade d-dimensional do espaço. Assumindo dois conjuntos de dados de cardinalidade N_{R_1} e N_{R_2}, com as respectivas aproximações MBR dos dados espaciais sendo armazenadas em dois índices de árvores R, R_1 e R_2.

Seja a altura da árvore $R_i (i = 1, 2)$ igual a h_{R_i} e assumindo que os nós raízes das árvores estão armazenados na memória. Para cada nível l_i, $1 \leq l_i \leq h_{R_i - 1}$, R_i contém N_{R_i, l_i} nós de tamanho médio s_{R_i, l_i}, sendo cada nó composto de um conjunto de entradas E_{R_i}, l_i. Seguindo o algoritmo SJ acima descrito, que implementa uma busca em profundidade,

as entradas E_{R_1,l_1} e E_{R_2,l_2} são comparadas para determinar os pares de entradas que estão sobrepostos. O custo da comparação para cada nível é a soma de dois fatores, que correspondem ao custo para as duas árvores $NA(R_1, R_2, l_1)$ e $NA(R_2, R_1, l_2)$. Esses custos podem ser calculados considerando que as entradas de R_1 (respectivamente R_2) representam um conjunto de dados (respectivamente, um conjunto de janelas de consulta q), utilizando a análise proposta para consultas de seleção *(selection queries)* do item 3.8.1, pode-se derivar os pares de nós de interseção para cada árvore R com o objetivo de estimar o custo de acesso.

Se não houver um esquema de *buffer*, o custo de acesso para ambas as árvores R_1 e R_2 no nível correspondente l_1 e l_2 é igual, já que um número igual de nós é acessado (linha 14 do algoritmo SJ). A linha 4 do algoritmo SJ é chamada repetidamente a cada nível das duas árvores descendo para o nível das folhas da árvore R_i. Assumiu-se que a árvore R_2 possui a menor altura. Para os $h_{R_2} - 1$ maiores nível das duas árvores:

$$NA(R_1, R_2, l_1) = NA(R_2, R_1, l_2) = \sum_{N_{R_2,l_2}} intersec(N_{R_1,l_1}, s_{R_1,l_1}, s_{R_2,l_2}) \Rightarrow \tag{3.52}$$

$$NA(R_1, R_2, l_1) = NA(R_2, R_1, l_2) = N_{R_2,l_2}.N_{R_1,l_1}.\prod_{k=1}^{d}(s_{R_1,l_1,k} + s_{R_2,l_2,k}), \tag{3.53}$$

onde $h_{R_1} - h_{R_2} + 1 \leq l_1 \leq h_{R_1} - 1$ e $1 \leq l_2 \leq h_{R_2} - 1$. Depois que os nós folhas da menor árvore R_2 tiverem sido acessados, l_2 fica associado ao valor fixo 1, denotando o nível folha, e a propagação de R_1 continua descendo por mais $h_{R_1} - h_{R_2}$ níveis.

$$NA\_total(R_1, R_2) = \sum_{l_1=1}^{h_{R_1}-1} \{NA(R_1, R_2, l_1) + NA(R_2, R_1, l_2)\}, \ onde \tag{3.54}$$

$$l_2 = \begin{cases} l_1 - (h_{R_1} - h_{R_2}), & h_{R_1} - h_{R_2} + 1 \leq l_1 \leq h_{R_1} - 1 \\ 1, & 1 \leq l_1 \leq h_{R_1} - h_{R_2} \end{cases} \tag{3.55}$$

A fórmula 3.54 é simétrica com respeito aos dois índices R_1 e R_2 refletindo a mesma situação existente no algoritmo SJ, onde o número de acessos a nós é igual ao número de chamadas à função ReadPage que por sua vez é igual para as duas árvores.

Capítulo 4

Dados e modelo de implementação

4.1 Dados

Suportar um grande volume de dados espaciais é uma característica inerente às aplicações modernas de banco de dados, como os *Geographical Information Systems (GIS), Computer Aided Design (CAD)* e *Image and Multimedia Databases* entre outros [36]. Por isso nosso estudo utilizou três tipos de dados com variados volumes : dados sintéticos simulando distribuição aleatória, dados sintéticos gerados a partir do modelo de distribuição de dados espaciais de Ciferri [6] e dados reais [12].

4.1.1 Dados sintéticos

Ultilizamos um conjunto de dados sintéticos nos testes de desempenho experimentais. O objetivo da utilização de dados sintéticos foi realizar uma investigação experimental mais abrangente do desempenho de técnicas de junção espacial baseadas em árvores-R. Em especial, a finalidade de se utilizar dados sintéticos é que estes dados permitem o controle do fator distribuição espacial dos dados, permitindo assim que se faça uma investigação do impacto deste fator no desempenho das técnicas de junção espacial. Também podemos configurar testes utilizando vários volumes diferentes de dados.

4.1.2 Dados sintéticos gerados por distribuição aleatória

Nosso estudo utilizou um primeiro conjunto de dados sintéticos de 2000, 4000, 8000, 16000, 32000, 64000 e 128000 objetos espaciais (retângulos com tamanho constante correspondendo a 2% do tamanho do espaço total de dados). Os dados foram gerados para o espaço $[0, 1)^2$. Esses dados sintéticos foram gerados com distribuição aleatória dos centróides e simulam uma distribuição uniforme de dados no *extent*. Foi utilizado um modelo com ob-

jetos quadrados e de de tamanho constante pois essas foram algumas premissas assumidas no modelo de Theodoridis [36], foi utilizada apenas um arquivo para cada volume de dados. Para que o modelo não fosse uma simulação exata de um distribuição uniforme, mas fosse uma simulaçao de dados reias, ao invés de utilizarmos uma distribuiçao uniforme dos centros das aproximações para cada eixo, utilizamos uma distribuição aleatória dos centros.

A tabela 4.1 apresenta a densidade espacial dos dados e os valores da capacidade média dos nós da etrutura de indexação árvore-R* CR para cada uma das configurações de *buffer* utilizada nos testes. Foi utilizada a árvore-R* CR pois este foi o método de indexação que apresentou o melhor desempenho segundo o estudo de métodos de acesso espacial realizado por Ciferri [6]. Na vertical temos a densidade do espaço e o valo da capacidade média dos nós para cada um dos tamanhos de pagina de disco testado (1k, 2k, 4k e 8k).

	Densidade	c (buff. 1k)	c (buff. 2k)	c (buff. 4k)	c (buff. 8k)
2k objetos	0,791187	0,669832	0,7039450	0,658117	0,604805
4k objetos	1,581305	0,675447	0,673219	0,670083	0,647844
8k objetos	3,164282	0,685307	0,682275	0,695112	0,684345
16k objetos	6,330481	0,673509	0,682316	0,682449	0,704190
32k objetos	12,668542	0,687503	0,687687	0,690356	0,684408
64k objetos	25,339997	0,675941	0,686928	0,691965	0,679642
128k objetos	50,694908	0,677673	0,686933	0,680543	0,688457

Tabela 4.1: Densidade e capacidade média c da árvore-R* para as configurações de *buffer*

No apêndice temos a visualização gráfica de alguns dos arquivos de dados gerados (ver figura (A.6) para arquivo de 2k objetos e figura (A.7) para arquivo de 4k objetos).

4.1.3 Dados sintéticos gerados segundo o modelo de Ciferri

Utilizamos um conjunto de dados sintéticos gerados segundo o modelo de distribuição espacial de dados definido em Ciferri [6]. Segundo Ciferri [6] o desempenho de um MAM é influenciado pela distribuição espacial dos dados que pode ser definida segundo modelos de localização coletiva de dados. Define-se um conjunto de possíveis modelos de distribuição de dados e os dados são gerados segundo esses modelos.

A distribuição espacial dos dados afeta diretamente o particionamento do espaço feito pela maioria dos MAMs influenciando na formação de regiões, o que interfere no agrupamento dos dados podendo degenerar a organização interna do MAM. O *extent* delimita

o espaço considerado nos testes de desempenho. Uma distribuição espacial define como os dados ocupam o *extent* indicando qual o relacionamento existente entre a localização espacial dos diversos dados, o que indica em quais partes do *extent* os dados estão localizados e como os dados estão organizados coletivamente nestas partes. Individualmente especifica a localização espacial de cada dado e coletivamente caracteriza a distribuição do conjunto de dados. O modelo gerador das distribuições abrange distribuições que diferem significativamente do ponto de vista coletivo, ou seja, diferem quanto à distribuição coletiva dos dados. Este modelo gera as distribuições segundo três passos (para mais detalhes ver Ciferri[6]):

- Definição da localização coletiva dos dados - Este passo caracteriza-se pela definição da subclasse de localização. Uma subclasse de localização caracteriza a localização relativa dos dados no *extent*, ou seja, a localização relativa dos dados em relação ao centro do *extent*, o formato e a área da região ocupada. Considera-se que duas regiões possuem uma localização relativa equivalente com relação ao centro do *extent* se a rotação do *extent* em torno do seu centro e/ou a inversão vertical ou horizontal do *extent* faz com que ambas as regiões passem a ter uma mesma localização absoluta. Define-se o conjunto (f, a, l) para caracterizar as distribuições, onde f indica o formato, a indica a área e l indica a localização relativa da região ocupada no *extent*. A ocupação relativa do *extent* é definida com base no conceito de quadrantes, o espaço é subdividido em quadrantes segundo o modelo *quadtree*. São definidos 5 tipos básicos de regiões: ocupação de apenas um quadrante, de dois quadrantes vizinhos, de dois quadrantes opostos, de três quadrantes e de quatro quadrantes. A subclasse de localização são especializadas na ocupação da parte central ou periférica do quadrante.

- Determinação da organização dos dados - Este passo caracteriza-se pela definição da classe de organização. A classe de organização indica a presença ou ausência de agrupamentos, e a ausência ou presença de pontos individuais.

- Aplicação da classe e subclasse de localização - Os pontos relativos à distribuição são gerados segundo a classe de organização escolhida dentro dos limites fixados pela subclasse de localização.

Os dados gerados por esse modelo e utilizados nos nossos testes constituiram-se de retângulos definidos no *extent* $[0, 1)^2$. Os centróides dos retângulos foram gerados de forma que estivessem contidos no limite da subclasse de localização e fossem agrupados segundo a sua classe de organização.

Nossos testes utilizaram quatro volumes de dados de 10000, 25000, 50000 e 100000 objetos. No apêndice temos a visualização gráfica de alguns dos arquivos de dados gerados

(ver figura (A.9) para o arquivo de dados D_SL20_COrg1_10k_retangulos_01 (10K_g1_01) e a figura (A.10) para o arquivo de dados D_SL20_COrg2_10k_retangulos_02 (10K_g2_02) e a figura (A.13) para o arquivo de dados D_SL20_COrg4_10k_retangulos_01 (10K_g4_01) e a figura (A.14)para o arquivo de dados D_SL20_COrg4_10k_retangulos_02 (10K_g4_02) no apêndice).

4.1.4 Dados reais

O conjunto de dados reais foi obtido junto ao projeto SAGRE Sistema Automatizado de Gerência de Rede Externa [23, 24, 22, 25], um sistema que utiliza dados geográficos, desenvolvido pelo Centro de Pesquisa e Desenvolvimento CPqD. O conjuntos de dados reais é relativo à cidade de Valinhos-SP. A função do sistema SAGRE é automatizar os processos relacionados ao cadastro, planejamento, projeto, implantação, manutenção, dentre outras operações, da rede externa das operadoras de telefonia. Esses dados também foram utilizados por Gatti [12] na sua análise.

No apêndice temos a visualização gráfica do arquivo de dados utilizado (ver figura (A.8) para arquivo de quadras no apêndice).

4.2 Modelo de implementação

Para realizar os testes experimentais utilizamos o modelo de implementação de Gatti [12]. No trabalho de Gatti [12] foi feito um estudo sobre os fatores que afetam o desempenho de junções espaciais. Estudou-se a influência dos fatores tamanho de página, do tamanho do *buffer-pool* e das políticas de substituição de páginas no desempenho dos algoritmos de junção espacial. O modelo de implementação utilizado possui uma arquitetura modularizada. O módulo gerenciador de blocos é o responsável pela interface entre o disco e os módulos de junção e de métodos de acesso, junto a este módulo foi implementado um *buffer-pool*. Os métodos de substituição de páginas estão implementados em um módulo separado, acessado apenas pelo Gerenciador de Blocos, o *Gerenciador de buffer-pool*, onde foi implementado como política de substituição de páginas o método *LRU (least recently used)*. Um fator configurável a partir de um parâmetro de entrada da operação é a quantidade de páginas a ser utilizada para *buffer pinning*. O *buffer pinning* consiste na fixação de páginas no *buffer-pool*, até que todas as referências a ela tenham sido processadas. Isto evita que uma página muito referenciada seja descartada pela política de substituição de páginas. O módulo de métodos de acesso espacial é o responsável pela inserção de elementos no índice, remoção, consulta e inicialização. Além disso, o método de acesso espacial mantém os parâmetros como tamanho do bloco utilizado, números máximo e mínimo de entradas permitidas em cada nó, número de dimensões, dentre outros parâmetros. O

módulo de métodos de junção espacial constitui-se de três métodos de junção espacial: o *nested loop* (NL) com entradas indexadas, o método *depth-first* (junção em profundidade DF) com as otimizações propostas por Brinkhoff [4], e o método *breadth-first* (junção em largura BF) proposto por Huang *etalii* [16]. Nós utilizaremos o mesmo modelo de modularização para a implementação. Para maiores detalhes de implementação do modelo experimental ver Gatti [12].

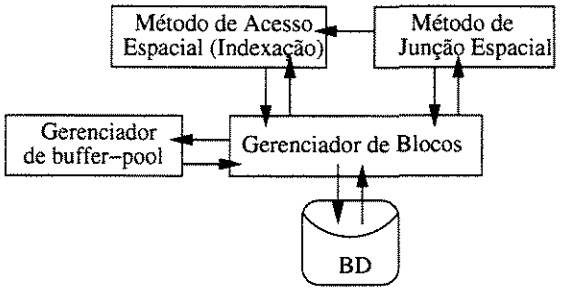


Figura 4.1: Arquitetura do sistema

Capítulo 5

Análise dos resultados

Os modelos analíticos possuem intrinsecamente uma margem grande de erro. Mas se este erro for consistente o modelo analítico atinge o seu objetivo. Portanto é importante testar vários casos para verificar a consistência. Em nossos testes utilizamos três tipos de dados: dados sintéticos gerados aleatoriamente para simular uma distribuição uniforme, dados sintéticos gerados segundo o modelo de distribuição de dados no espaço proposto por Ciferri [6] e dados reais. Os dados reais foram os mesmos utilizados por Gatti [12], por isso nossa pesquisa não enfocou os dados reais, já que os resultados provenientes das junções com dados reais podem ser obtidos do trabalho de Gatti [12].

Para cada um dos conjuntos de dados sintéticos utilizamos variados volumes de dados para poder verificar qual a influência do fator volume de dados nos valores obtidos pelos métodos analíticos para junções espaciais.

5.1 Métodos analíticos

Escolhemos o método analítico de Theodoridis [36] para a nossa análise porque esse é o método para junções espaciais mais recente presente na literatura para o escopo do trabalho. Além disso este método baseia-se apenas nas propriedades dos dados, o que o torna um método simples e de baixo custo de aplicação.

No levantamento sobre métodos analíticos constatamos que o modelo analítico de Yun-Wu Huang e Ning Jing [15] apresentava um comportamento muito insatisfatório. Ao estudar o modelo analítico proposto neste trabalho [15], bem como outros trabalhos correlatos mais recentes presentes na literatura [20, 15], constatamos que o modelo precisava de um ajuste. Propomos uma correção ao modelo [15].

O modelo probabilístico para prever o número médio de retângulos interceptados por um retângulo r foi inicialmente proposto em [20] da seguinte forma:

Lema. Dado um conjunto de N retângulos $r_1, ..., r_N$ com tamanho médio s e um

retângulo r com tamanho q, o número médio de retângulos interceptados por r é:

$$intersec(N, s, q) = N. \prod_{k=1}^{d} (s_k + q_k) \tag{5.1}$$

Prova. Dado um conjunto de N retângulos com tamanho médio s e um retângulo r com tamanho q, o número médio de retângulos deste conjunto que interceptam o retângulo r é igual ao número de retângulos de um segundo conjunto de N retângulos de tamanho médio s' que contém um ponto no espaço, onde $s'_K = s_k + q_k, \forall k$. Por definição, está é a densidade D' do segundo conjunto de retângulos.

Posteriormente à apresentação dos trabalhos [20, 15] o trabalho [21] criticou esse modelo inicial de probabilidade de interseção, propondo uma correção para o modelo [20]. Este modelo [20] pode gerar valores maiores que 1 para as probabilidades de interseção entre dois retângulos , o que não tem sentido.

Para ilustrar os problemas deste modelo probabilístico [20] tomemos um exemplo. Seja $d = 2$ a dimensão do espaço de dados e $WS = [0, 1)^d$ a unidade dimensional do espaço, isto é, a área total do espaço é 1. Suponha um conjunto de apenas 1 retângulo ($N = 1$) com tamanho $s = 0.99$ para cada uma das dimensões e uma janela de consulta r com tamanho $q = 0.98$ para cada uma das dimensões. Utilizando o modelo proposto em [20], a probabilidade de interseção entre essas duas entradas é:

$intersec(N, s, q) = N. \prod_{k=1}^{d} (s_k + q_k) =$
$= 1 * ((0.99 + 0.98) * (0.99 + 0.98)) = 3.94!!!$

Para contornar esse problema o trabalho [36] utilizou um modelo diferente para calcular a probabilidade de interseção:

$$intersec(N, s, q) = N. \prod_{k=1}^{d} min\{1, (s_k + q_k)\}$$

Com isso a probabilidade de dois retângulos terem interseção fica limitada a um valor máximo 1. Para a fórmula inicialmente proposta para o modelo [15]:

$ZEIO_h = 2 + 2 \times \sum_{l=1}^{h-1} \sum_{i=1}^{N^l} \sum_{j=1}^{\tilde{N}^l} (x_i^l + \tilde{x}_i^l) \times (y_i^l + \tilde{y}_j^l)$

Propomos utilizar o mesmo modelo de probabilidade proposto em [36] para corrigir a fórmula de probabilidade do modelo analítico [15]. Com isso, a nova fórmula de $ZEIO_h$ para o modelo [15] pode ser assim definida:

$$2 + 2 \times \sum_{l=1}^{h-1} \sum_{i=1}^{N^l} \sum_{j=1}^{\tilde{N}^l} min\{(x_i^l + \tilde{x}_i^l), 1\} \times min\{(y_i^l + \tilde{y}_j^l), 1\} \tag{5.2}$$

5.2 Ambiente de teste

Os testes de desempenho foram medidos considerando o número de acessos a disco (E/S) para as operações de junção espacial. Foi utilizada uma máquina *PC* (Processador *Pentium II*) com 160 MB de memória *RAM* e um de disco 15GB rodando o sistema operacional linux. A análise analítica ou não analítica de métodos envolve dois tipos de complexidade: a complexidade de tempo e a complexidade de espaço. A primeira está relacionada com o tempo demandado pelo método para a sua execução. A segunda envolve os requisitos de espaço do método, seja em memória principal ou em disco.

Para a análise de desempenho escolhemos estudar o fator número de acessos a disco. A ênfase deste experimento se dará em relação aos requisitos de tempo. Isto se deve ao fato de que, em sistemas interativos como sistemas gerenciadores de bancos de dados (SGBDs) ou sistemas de informações geográficas (SIGs), um dos requisitos de grande satisfação do usuário é o *tempo* que determinada operação leva até sua conclusão. Isto sem considerar que, quanto mais rapidamente uma operação for executada, mais rapidamente o processo que a requisitou liberará os recursos a ele alocados [12].

Operações típicas de SGBDs e SIGs envolvem grandes quantidades de acessos a disco. Estas operações de I/O são extremamente caras se comparadas às instruções executadas pela CPU. Enquanto estas estão na casa dos milissegundos, aquelas estão na faixa dos microssegundos. Se considerarmos que uma operação de junção ou outra consulta qualquer pode acessar todas as páginas de determinado índice, e que este índice pode compreender milhares de páginas, uma execução ineficiente demandará altos custos em relação ao tempo de execução dessa consulta [12].

5.3 Critérios para o modelo de implementação

A implementação definida em Gatti [12] foi utilizada rodando com o conjunto de dados descritos no capítulo anterior. Para padronizar a comparação dos resultados da implementação, utilizamos sempre a busca em profundidade, sem fixação de páginas de *buffer* e durante o processamento da junção os objetos espaciais foram ordenados segundo o valor de *Hilbert* do seu centróide *(space-filling curve)*. O tamanho do *buffer* variou em uma faixa de 1k, 2k, 4k e 8k. A política de gerenciamento de páginas no *buffer* utilizada foi à política *LRU*. A indexação dos objetos espaciais para a junção utilizou a estrutura da árvore-R*, pois essa foi à estrutura considerada nos testes de Theodoridis [36]. Nosso objetivo é verificar os métodos propostos por Theodoridis [36] e confrontar os resultados obtidos no trabalho de Theodoridis [36] com os nossos resultados. Utilizamos a variação árvore-R* CR.

5.4 Análise dos resultados

Os resultados obtidos por nossa pesquisa aparecem nas tabelas a seguir. O trabalho de Theodoridis [36] chegou a resultados diferentes dos nossos. Atribuímos essa diferença a diferença nos dados utilizados. No trabalho de Theodoridis [36] os dados espaciais foram indexados por uma estrutura baseada na árvore-R*, com ocupação média de nós $c = 67\%$ e os resultados obtidos apresentaram um erro relativo entre 10% e 20% entre o experimental e o analítico. O nosso trabalho utilizou a mesma estrutura de indexação de Theodoridis [36].

Para os nossos dados sintéticos, a diferença entre os resultados de desempenho verificados nos testes experimentais e os resultados de desempenho gerados pelos modelos analíticos alcançou até aproximadamente 30%. As tabelas de 2k objetos (tabela 5.1 e gráfico 5.1), 4k objetos (tabela 5.2 e gráfico 5.2), 8k objetos (tabela 5.3 e gráfico 5.3), 16k objetos (tabela 5.4 e gráfico A.1), 32k objetos (tabela 5.5 e gráfico A.2), 64k objetos (tabela 5.6 e gráfico A.3) e 128k objetos (tabela 5.7 e gráfico A.4) ilustram os resultados obtidos.

Ao utilizar os dados gerados a partir do modelo de distribuição de Ciferri [6], observamos que os valores analíticos não têm um comportamento satisfatório. As tabelas para 10k objetos (tabela 5.8 e gráfico 5.4), 25k objetos (tabela 5.9 e gráfico 5.5), 50k ojbetos (tabela 5.10) e 100k objetos (tabela 5.11 e gráfico A.5) ilustram os resultados obtidos por essas distribuições. Os arquivos de dados gerados segundo o modelo de Ciferri [6] possuem objetos que não são quadrados, mas retângulos. O modelo de Theodoridis [36] assume a premissa, para a definição das fórmulas matemáticas, que os objetos espaciais são quadrados. Além disso o modelo de Theodoridis [36] também assume uma densidade uniforme do espaço. Podemos verificar pelas distribuições de dados utilizadas que os dados não apresentam uma densidade constante no espaço (ver no apêndice as figuras (A.9) para o arquivo de dados D_SL20_COrg1_10k_retangulos_01 (10K_g1_01) e a figura (A.10) para o arquivo de dados D_SL20_COrg2_10k_retangulos_02 (10K_g2_02) e a figura (A.13) para o arquivo de dados D_SL20_COrg4_10k_retangulos_01 (10K_g4_01) e a figura (A.14)para o arquivo de dados D_SL20_COrg4_10k_retangulos_02 (10K_g4_02)).

O resultado de dados real foi obtido a partir da junção de arquivo de dados de quadras [12] e aparece na tabela 5.5.

De um modo geral, podemos observar que :

- quanto mais não uniforme for a distribuição de dados maior a diferença entre o valor analítico e o valor experimental. Isto pode ser explicado pelo fato do modelo analítico de Theodoridis [36] utilizar premissas para a elaboração do modelo analítico que caracteriza uma distribuição uniforme de dados. Uma dessas premissas é a utilização da densidade do espaço como um parâmetro para refletir as densidades

das subáreas do espaço. Esse problema foi abordado por Theodoridis [36], mas foi apresentada apenas uma solução para o caso de consultas de seleção. Para as consultas de seleção (consultas simples por janela) Theodoridis [36] propõe que não seja utilizada a densidade global do espaço para a análise, mas se mantenha um gride do espaço, onde seja guardada uma densidade por gride, então a localização da janela de consulta determinará qual o valor de densidade a ser utilizada. Para junções não é possível utilizar essa solução, visto que não é possível, seguindo o modelo de Theodoridis [36], determinar localizações da janela de consulta para junção, ou seja, não é possível utilizar o fato da densidade não ser constante no espaço. As tabelas de 2k objetos (tabela 5.1), 4k objetos (tabela 5.2), 8k objetos (tabela 5.3), 16k objetos (tabela 5.4), 32k objetos (tabela 5.5), 64k objetos (tabela 5.6) e 128k objetos (tabela 5.7) ilustram os resultados obtidos para as simulações de distribuições uniformes, podemos observer que a diferença entre os valores analíticos e sintético para essas tabelas e menor e seu comportamento e mais uniforme do que o comportamento dos resultados apresentados pelas distribuições não uniformes presente nas tabelas para 10k objetos (tabela 5.8 , 25k objetos (tabela 5.9, 50k ojbetos (tabela 5.10) e 100k objetos (tabela 5.12) para distribuições não uniformes.

- Para os dados de distribuição não-uniforme que seguem o modelo de distribuição espacial de dados de Ciferri [6], resaltamos o impacto do fator cluster como um fator que degrada bastante o valor obtido pelo método analítico, ou seja, um fator que gera valores experimentais e analíticos com grande diferença. Essa característica de interferência do fator cluster pode ser notada na quinta e sexta coluna das tabelas 5.8 e 5.10 e na quarta e quinta colunas da tabela 5.9 e na terceira e quarta colunas da tabela 5.12. Nessas colunas destas tabelas estão sendo ilustrados os resultados de uma junção onde pelo menos um dos conjuntos de dados possuem dados cuja distribuição espacial apresenta a clusterização.

O modelo analítico de Theodoridis [36] tem a vantagem de ser um modelo que depende apenas das propriedades dos dados (densidade do espaço e número de objetos) e uma única propriedade de fácil obtenção da estrutura de indexação , a ocupação média dos nós da árvore. Mas essa simplicidade do modelo foi alcançada definindo-se um conjunto de premissas muito restritivas quanto ao modelo de dados. Esse modelo pode ser aplicado independente do modelo de distribuição de dados apenas para as consultas por janela, onde a densidade utilizada na análise reflete a densidade da janela. Para as junções espaciais esse modelo apresenta-se adequado apenas para as distribuições uniformes de dados, onde a densidade de uma região pode ser diretamente derivada da densidade global do espaço, para distribuições sem uniformidade o modelo analítico não apresenta um resultado satisfatório. Verificamos a necessidade de adaptar o modelo de Theodoridis [36]

de forma a abordar os casos onde a densidade do espaço não é constante.

Um outro parâmetro utilizado por Theodoridis [36] é a ocupação média dos nós na árvore de indexação (c). Considera-se para efeito de análise que esse valor médio é o valor da ocupação para cada nó na árvore. Entretanto esse é apenas um valor médio e dependendo dos formatos e distribuição dos dados espaciais, esse valor médio pode não ser uma boa aproximação para o valor da ocupação para os nós da árvore de indexação. Isto leva a determinação errada do número de entradas por nós e conseqüentemente a um erro na previsão do número de acessos a disco para a junção.

	2k	4k	8k	16k	32k	64k	128k
anal.(buff. 1k)	888	1379	2641	4416	7502	13663	25611
exp.(buff. 1k)	1122	1830	2916	4063	7746	13988	25839
analit./exp	0,79144	0,75355	0,90569	1,08688	0,9685	0,97677	0,99118
anal.(buff. 2k)	402	634	1023	1741	3063	5903	10859
exp.(buff. 2k)	550	649	1093	1904	3618	6749	27213
analit./exp	0,7301	0,97689	0,93596	0,914391	0,8466	0,874648	0,399037
anal.(buff. 4k)	215	322	576	946	1585	2792	5174
exp.(buff. 4k)	268	426	650	1020	1882	3497	6622
analit./exp.	0,80224	0,75587	0,88615	0,92745	0,84219	0,7984	0,78134
anal.(buff. 8k)	117	168	255	412	776	1366	2452
exp.(buff. 8k)	142	204	292	462	778	1702	3195
analit./exp	0,82394	0,82353	0,87329	0,89178	0,98458	0,80259	0,76745

Tabela 5.1: Junção entre o arquivo de 2k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

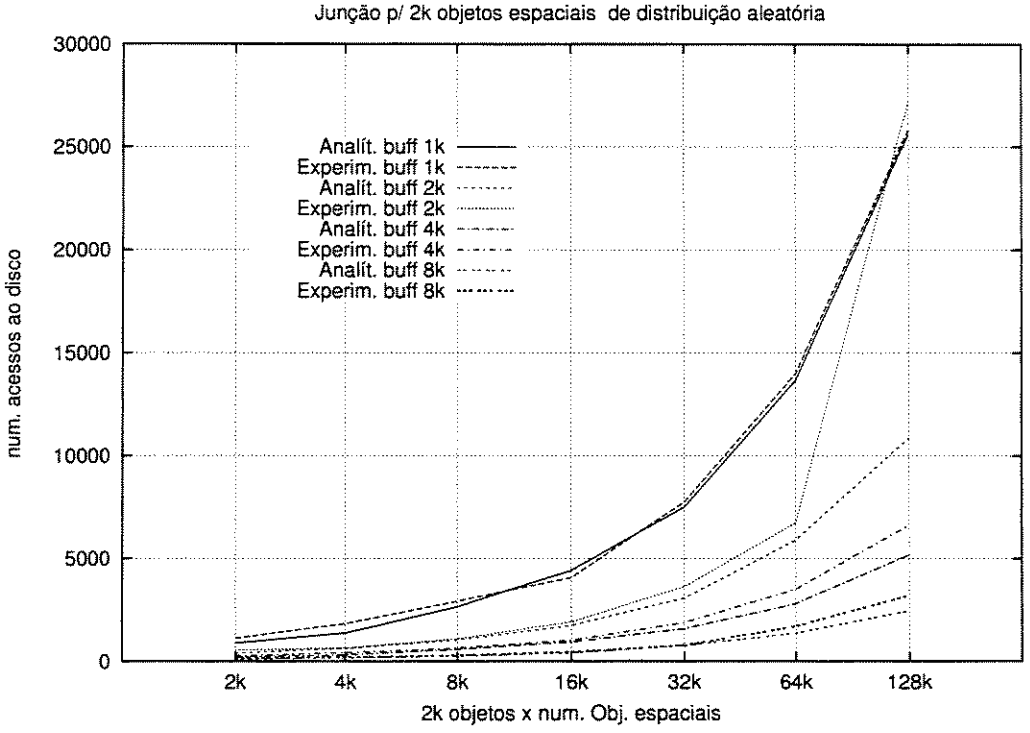


Figura 5.1: Gráfico da juunção entre o arquivo de 2k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

	2k	4k	8k	16k	32k	64k	128k
anal.(buff. 1k)	1379	2072	4402	6461	10682	19020	35265
exp.(buff. 1k)	1830	2908	4426	6789	12446	23966	44824
analit./exp.	0,75355	0,71252	0,9042	0,95169	0,85827	0,79362	0,78674
anal.(buff. 2k)	634	956	1484	2432	4143	7118	14233
exp.(buff. 2k)	683	1342	2054	3246	5388	9484	21614
analit./exp.	0,92826	0,71237	0,72249	0,74923	0,76893	0,75053	0,65851
anal.(buff. 4k)	322	461	837	1294	2083	3553	6419
exp.(buff. 4k)	426	662	966	1643	3032	5642	11024
analit./exp.	0,75587	0,69638	0,8665	0,78758	0,68701	0,62974	0,58228
anal.(buff. 8k)	168	232	339	528	975	1674	2928
exp.(buff. 8k)	204	304	430	640	1036	2840	5382
analit./exp.	0,82353	0,76316	0,78837	0,825	0,94112	0,58944	0,54404

Tabela 5.2: Junção entre o arquivo de 4k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

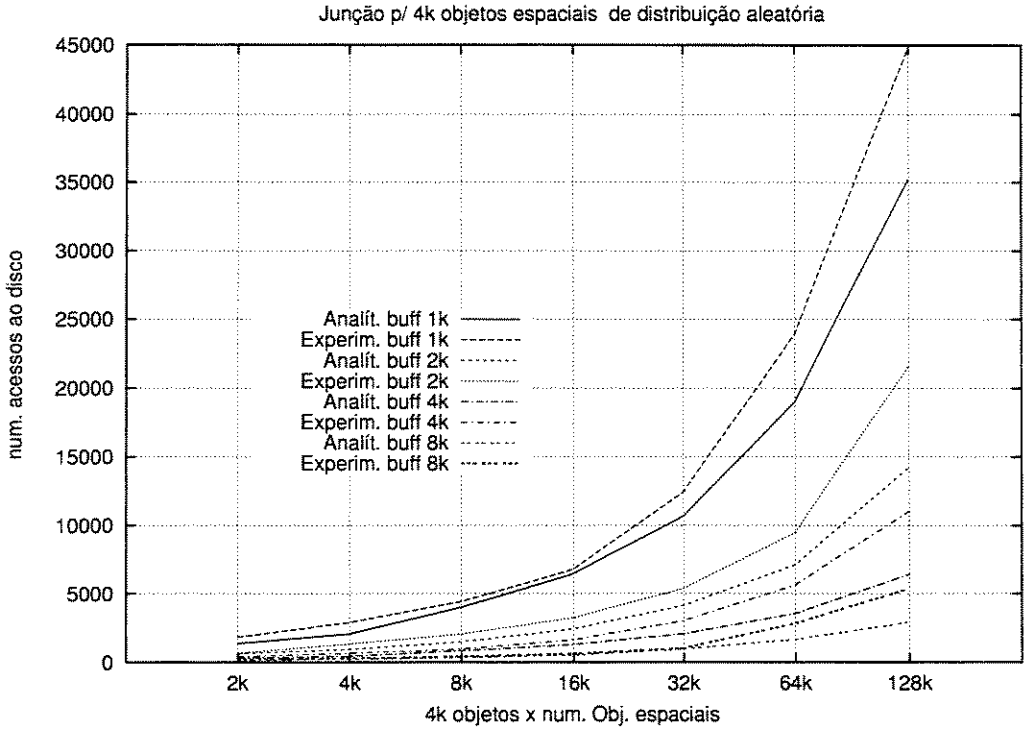


Figura 5.2: Gráfico da junção entre o arquivo de 4k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

	2k	4k	8k	16k	32k	64k	128k
anal.(buff. 1k)	2641	4002	5031	8312	14004	25216	49969
exp.(buff. 1k)	2916	4426	6822	11471	21467	41954	79378
analit./exp.	0,905569	0,9042	0,73747	0,72461	0,65235	0,60104	0,62958
anal.(buff. 2k)	1023	1487	2214	3513	5825	11099	19508
exp.(buff. 2k)	1136	2054	3110	4736	7470	12708	38352
analit./exp.	0,90053	0,72395	0,7119	0,74177	0,77979	0,87339	0,50866
anal.(buff. 4k)	576	837	997	1567	2548	4362	7874
exp.(buff. 4k)	650	966	1436	2703	5106	9707	19403
analit./exp.	0,88615	0,86646	0,6943	0,57973	0,49902	0,44937	0,40581
anal.(buff. 8k)	255	399	476	714	1317	2160	3665
exp.(buff. 8k)	292	430	634	946	1418	4892	9396
analit./exp.	0,87329	0,92791	0,75079	0,75476	0,92877	0,44154	0,39006

Tabela 5.3: Junção entre o arquivo de 8k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

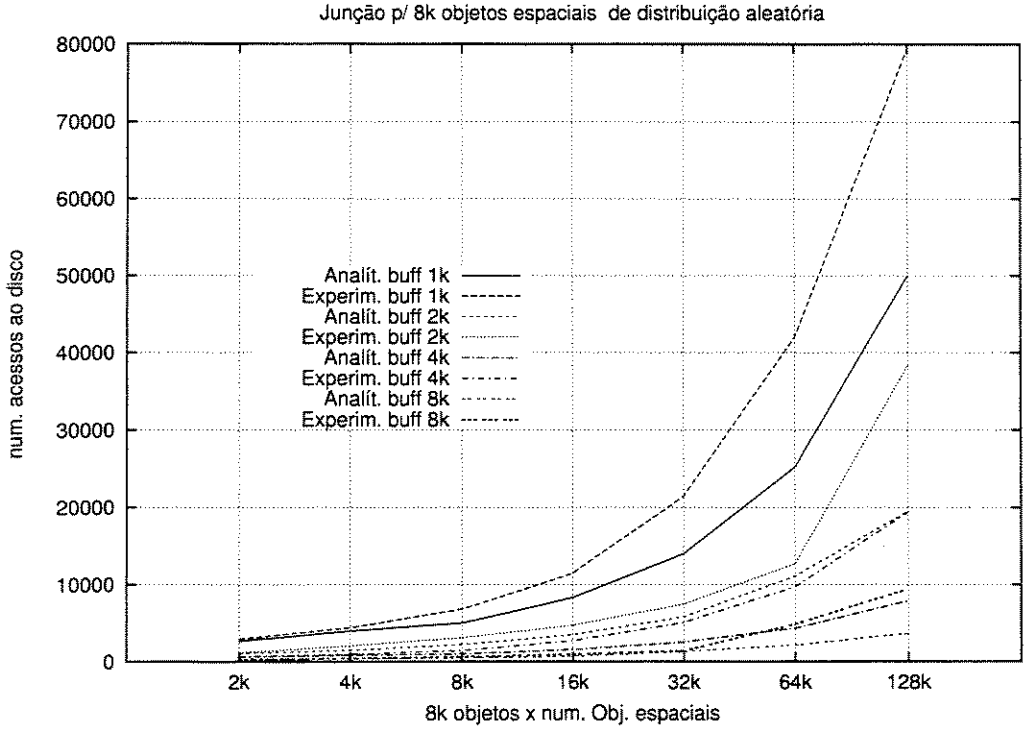


Figura 5.3: Gráfico da junção entre o arquivo de 8k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

	2k	4k	8k	16k	32k	64k	128k
anal.(buff. 1k)	4416	6461	8257	13402	22151	39212	77855
exp.(buff. 1k)	4168	7015	11955	16994	26984	47468	86594
analit./exp.	1,0595	0,92103	0,69067	0,78863	0,82089	0,82607	0,89908
anal.(buff. 2k)	1741	2432	3513	5406	8733	16610	28524
exp.(buff. 2k)	1971	3246	4736	7304	11096	18010	70141
analit./exp.	0,883308	0,74923	0,74177	0,74014	0,78704	0,92227	0,40667
anal.(buff. 4k)	946	1294	1567	2370	3731	6196	10898
exp.(buff. 4k)	1057	1721	2848	3296	4806	7320	12478
analit./exp.	0,89499	0,75189	0,55021	0,71905	0,77632	0,84645	0,87338
anal.(buff. 8k)	412	528	714	1029	1893	2970	4872
exp.(buff. 8k)	462	640	946	1426	2074	8541	16890
analit./exp.	0,89177	0,825	0,75476	0,7216	0,91273	0,34773	0,28846

Tabela 5.4: Junção entre o arquivo de 16k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

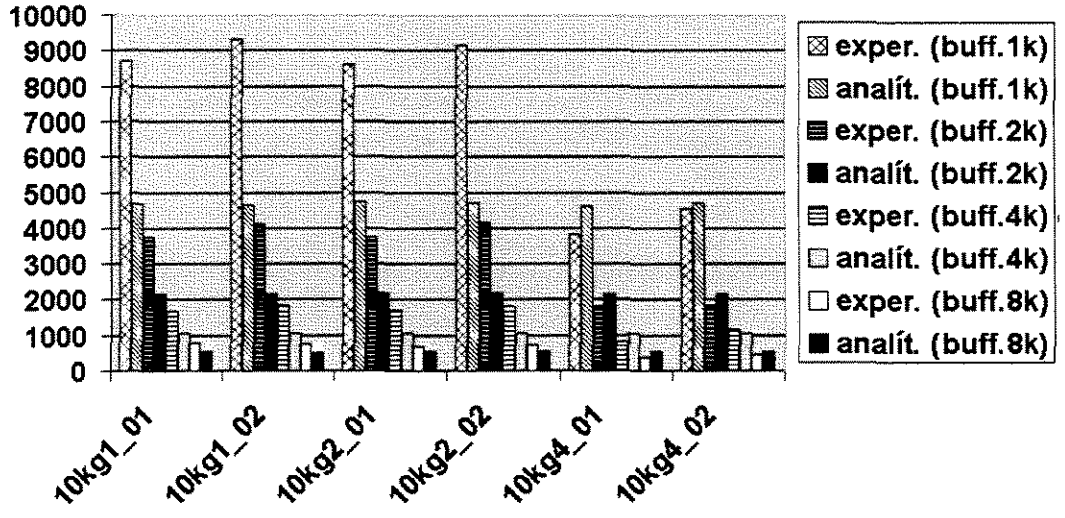


Figura 5.4: Gráfico da junção entre o arquivo 10k_g1_01 e os arquivos 10k_g1_01, 10k_g1_02, 10k_g2_01, 10k_g2_02, 10k_g4_01 e 10k_g4_02 (Dados sintéticos de 10k objetos simulando distribuição não uniforme segundo o modelo de distribuição de dados de ciferri)

	2k	4k	8k	16k	32k	64k	128k
anal.(buff. 1k)	7502	10682	14004	21151	36047	62916	125232
exp.(buff. 1k)	7601	12762	22115	26984	43148	74684	134392
analit./exp.	0,98698	0,83702	0,63324	0,78384	0,83543	0,84243	0,93184
anal.(buff. 2k)	3063	4143	5825	8733	13778	26195	44009
exp.(buff. 2k)	3698	5388	7470	11096	17070	27352	132085
analit./exp.	0,82829	0,76893	0,77979	0,78704	0,80715	0,9577	0,33319
anal.(buff. 4k)	1585	2083	2548	3731	5689	9191	15792
exp.(buff. 4k)	1928	3140	5325	4806	7262	10882	17810
analit./exp.	0,8221	0,66338	0,4785	0,77632	0,78339	0,84461	0,88669
anal.(buff. 8k)	766	975	1317	1893	2461	3906	6432
exp.(buff. 8k)	778	1036	1418	2074	3176	7851	18471
analit./exp.	0,98458	0,94112	0,92877	0,91273	0,77487	0,34822	0,34822

Tabela 5.5: Junção entre o arquivo de 32k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

	2k	4k	8k	16k	32k	64k	128k
anal.(buff. 1k)	13663	19020	25216	39212	62916	108433	216223
exp.(buff. 1k)	14148	28368	42731	47468	74684	127660	224868
analit./exp.	0,96572	0,670474	0,59011	0,826072	0,842429	0,84939	0,96155
anal.(buff. 2k)	5903	7918	11099	16610	26195	37618	64427
exp.(buff. 2k)	6846	9484	12424	18010	27352	44000	254112
analit./exp.	0,86226	0,83488	0,89335	0,92227	0,9577	0,85496	0,253538
anal.(buff. 4k)	2792	3553	4362	6196	9191	14483	24339
exp.(buff. 4k)	3555	5776	9997	7320	10882	16672	27166
analit./exp.	0,78537	0,61513	0,43633	0,84645	0,84461	0,86870	0,89594
anal.(buff. 8k)	1366	1674	2160	2970	3906	6004	9618
exp.(buff. 8k)	1738	2915	5062	8895	7936	6990	10092
analit./exp.	0,78596	0,57427	0,42671	0,3339	0,49219	0,85894	0,95303

Tabela 5.6: Junção entre o arquivo de 64k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

	2k	4k	8k	16k	32k	64k	128k
anal.(buff. 1k)	25611	35265	49968	77855	125232	216223	336314
exp.(buff. 1k)	26020	45220	80284	86594	20448	224868	391892
analit./exp.	0,98428	0,77985	0,62239	0,89908	1,00612	0,996156	0,85818
anal.(buff. 2k)	10859	14233	18508	28524	44009	64427	108924
exp.(buff. 2k)	27213	21866	38898	71248	134255	258513	130404
analit./exp.	0,39904	0,65092	0,47581	0,40035	0,3278	0,24922	0,83528
anal.(buff. 4k)	5174	6419	7874	10898	15792	24339	40088
exp.(buff. 4k)	8718	14246	22261	12478	17810	27166	44582
analit./exp.	0,59348	0,45058	0,35371	0,87338	0,88669	0,89594	0,8992
anal.(buff. 8k)	2452	2928	3665	4872	6432	9618	15023
exp.(buff. 8k)	4207	6423	9411	13252	18660	10092	15230
analit./exp.	0,58284	0,45586	0,38944	0,36764	0,34469	0,95303	0,98648

Tabela 5.7: Junção entre o arquivo de 4k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

	10k_g1_01	10k_g1_02	10k_g2_01	10k_g2_02	10k_g4_01	10k_g4_02
anal.(buff. 1k)	4672	4627	4734	4713	4393	4714
exp.(buff. 1k)	8696	9396	8606	9140	3808	4550
analit./exp.	0,53726	0,49721	0,55008	0,51565	1,20615	1,03604
anal.(buff. 2k)	2149	2139	2172	2194	2153	2165
exp.(buff. 2k)	3750	4114	3754	4164	1794	1830
analit./exp.	0,57307	0,51993	0,57858	0,5269	0,1,20011	1,1830
anal.(buff. 4k)	1058	1043	1064	1061	1069	1069
exp.(buff. 4k)	1688	1834	1716	1794	810	1148
analit./exp.	0,62678	0,5687	0,62005	0,59142	1,31975	0,93119
anal.(buff. 8k)	530	521	540	535	547	551
exp.(buff. 8k)	770	758	670	704	380	452
analit./exp.	0,62678	0,56870	0,62005	0,59142	1,31975	0,93119

Tabela 5.8: Junção entre o arquivo 10k_g1_01 e os arquivos 10k_g1_01, 10k_g1_02, 10k_g2_01, 10k_g2_02, 10k_g4_01 e 10k_g4_02 (Dados sintéticos de 10k objetos simulando distribuição não uniforme segundo o modelo de distribuição de dados de ciferri)

	25k_g1_01	25k_g1_02	25k_g2_02	25k_g4_01	25k_g4_02
anal.(buff. 1k)	14882	14954	15140	15208	14777
exp.(buff. 1k)	39492	40586	39464	20828	2982
analit./exp.	0,376836	0,368452	0,383641	0,730171	4,955399
anal.(buff. 2k)	6460	6434	6500	6602	6367
exp.(buff. 2k)	16618	16760	16248	8660	6408
analit./exp.	0,388735	0,38389	0,400049	0,762356	0,993602
anal.(buff. 4k)	2999	3037	3073	3079	2977
exp.(buff. 4k)	7082	7196	7218	3802	3160
analit./exp.	0,423468	0,42204	0,425741	0,809837	0,942089
anal.(buff. 8k)	1413	1431	1423	1460	1444
exp.(buff. 8k)	2886	2964	2760	1630	1284
analit./exp.	0,489605	0,482794	0,51558	0,895706	1,24611

Tabela 5.9: Junção entre o arquivo 25k_g1_01 e os arquivos 25k_g1_01, 25k_g1_02, 25k_g2_02, 25k_g4_01 e 25k_g4_02 (Dados sintéticos de 25k objetos simulando distribuição não uniforme segundo o modelo de distribuição de dados de ciferri)

	50k_g1_01	50k_g1_02	50k_g2_01	50k_g2_02	50k_g4_01	50k_g4_02
anal.(buff. 1k)	37939	38229	39118	39137	37672	38430
exp.(buff. 1k)	124610	130242	124804	124700	67172	66050
analit./exp.	0,30446	0,29352	0,31343	0,31385	0,56083	0,58183
anal.(buff. 2k)	15449	15522	15752	15709	15263	15589
exp.(buff. 2k)	49238	50102	50588	48786	25166	25708
analit./exp.	0,31376	0,30981	0,31138	0,32199	0,60649	0,60639
anal.(buff. 4k)	7040	7021	7134	7067	6942	7065
exp.(buff. 4k)	21480	22062	21332	21594	11224	11390
analit./exp.	0,327747	0,31824	0,334427	0,327267	0,618496	0,620281
anal.(buff. 8k)	3173	3156	3176	3170	3176	3194
exp.(buff. 8k)	8676	8698	8680	8420	4502	4472
analit./exp.	0,365722	0,362842	0,365899	0,376487	0,705464	0,714222

Tabela 5.10: Junção entre o arquivo 50k_g1_01 e os arquivos 50k_g1_01, 50k_g1_02, 50k_g2_01, 50k_g2_02, 50k_g4_01 e 50k_g4_02 (Dados sintéticos de 50k objetos simulando distribuição não uniforme segundo o modelo de distribuição de dados de ciferri)

	100k_g1_01	100k_g1_02	100k_g2_01	100k_g2_02
anal.(buff. 1k)	105637	128097	107864	106479
exp.(buff. 1k)	419670	420104	394236	385260
analit./exp.(buff. 1k)	0,251774	0,304917	0,273603	0,276123
anal.(buff. 2k)	101094	31717	40474	39791
exp.(buff. 2k)	159248	160020	161668	157600
analit./exp.(buff. 2k)	0,634821	0,2482	0,250353	0,252481
anal.(buff. 4k)	16221	16701	16953	16811
exp.(buff. 4k)	63988	65320	63394	60116
analit./exp.(buff. 4k)	0,253501	0,25568	0,267423	0,279643
anal.(buff. 8k)	7225	7212	7482	7236
exp.(buff. 8k)	25102	24954	27292	24734
analit./exp.(buff. 8k)	0,287826	0,289012	0,274146	0,292553

Tabela 5.11: Junção entre o arquivo 100k_g1_01 e os arquivos 100k_g1_01, 100k_g1_02, 100k_g2_01 e 100k_g2_02 (Dados sintéticos de 100k objetos simulando distribuição não uniforme segundo o modelo de distribuição de dados de ciferri)

	100k_g4_01	100k_g4_02	21_100k_g1_01	21_100k_g2_02
anal.(buff. 1k)	109398	111319	106337	110069
exp.(buff. 1k)	231430	204858	138434	259368
analit./exp.(buff. 1k)	0,472704	0,543396	0,76814219	0,424373863
anal.(buff. 2k)	40818	41254	39922	41066
exp.(buff. 2k)	92550	82662	147304	107532
analit./exp.(buff. 2k)	0,441037	0,499068	0,271017759	0,381895622
anal.(buff. 4k)	17032	17332	16790	17129
exp.(buff. 4k)	31234	31448	62980	46310
analit./exp.(buff. 4k)	0,536711	0,551132	0,266592569	0,369876916
anal.(buff. 8k)	7275	7448	7279	7326
exp.(buff. 8k)	13444	14848	24522	20964
analit./exp.(buff. 8k)	0,541134	0,501616	0,296835495	0,354015657

Tabela 5.12: Cont. Junção entre o arquivo 100k_g1_01 e os arquivos 100k_g4_01, 100k_g4_02, 21_100k_g1_01 e 21_100k_g2_02 (Dados sintéticos de 100k objetos simulando distribuição não uniforme segundo o modelo de distribuição de dados de ciferri)

	1k	2k	4k	8k
analítico	1167	557	257	134
experimental	2204	912	320	190
analit./exp.	0,529492	0,610746	0,803125	0,705263

Tabela 5.13: Quadras x Quadras

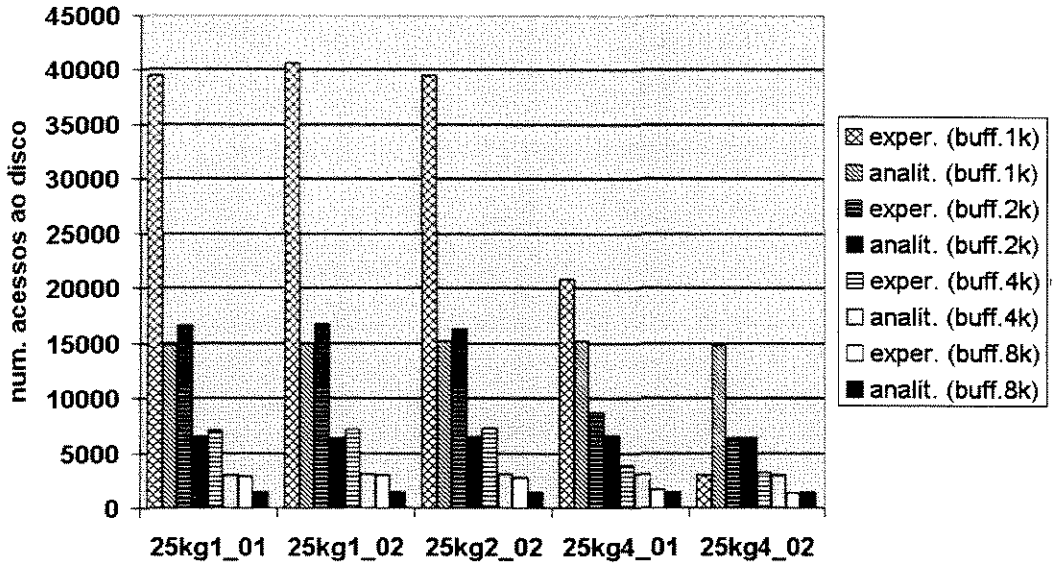


Figura 5.5: Gráfico da junção entre o arquivo 25k_g1_01 e os arquivos 25k_g1_01, 25k_g1_02, 25k_g2_02, 25k_g4_01 e 25k_g4_02 (Dados sintéticos de 25k objetos simulando distribuição não uniforme segundo o modelo de distribuição de dados de ciferri)

Capítulo 6

Conclusões

Como uma extensão ao trabalho de Gatti [12] e utilizando a implementação apresentada em Gatti [12] fizemos um estudo comparativo entre resultados experimentais e o método analítico de Theodoridis [36] para junções espaciais. Utilizamos diferentes conjuntos de dados sintéticos. Parte desses dados foram obtidas do modelo de distribuição de dados definido por Ciferri [6].

Nosso objetivo foi avaliar os métodos analíticos para junções espaciais baseando-se no fator de controle distribuição espacial de dados e volume de dados. Objetivamos avaliar os modelos analíticos atualmente propostos, buscando uma correlação entre a distribuição espacial dos dados e os fatores que determinam o desempenho calculado pelos modelos analíticos.

Um primeiro resultado relevante baseado nos resultados obtidos através de nossos testes preliminares foi à alteração que foi proposta ao modelo analítico de Yun-Wu Huang e Ning Jing [15].

Os modelos analíticos assumem algumas premissas para definição de suas fórmulas básicas, como a premissa de que as entradas na árvore de indexação são quadrados ou os centros das projeções dos retângulos estão igualmente distanciados para as entradas da árvore. Muitas destas características definem um tipo básico de distribuição espacial, como a distribuição uniforme. Além disto, o modelo analítico utilizado em nossa pesquisa [36] utiliza a densidade do espaço com um parâmetro para o cálculo do número de acessos a disco, mas essa densidade pode não ser uniforme no espaço, o que gera um fator de fragilidade nos resultados obtidos através dos métodos analíticos. Seria importante definir novas fórmulas que de alguma forma refletissem a ocupação desigual do espaço de dados. Os modelos analíticos estudados têm uma boa aplicação para dados de distribuição uniforme, nossa pesquisa concluiu a necessidade de adaptar os métodos analíticos para junções espaciais às características peculiares dos dados de uma aplicação espacial com distribuições mais genéricas de dados.

6.1 Extensões

Podemos propor como extensões a esse trabalho:

- Adaptar o modelo analítico para quaisquer distribuições espaciais de dados.
- Comparar os resultados obtidos ampliando o modelo original e a correção proposta ao modelo de Yun-Wu Huang e Ning Jing [15]
- Analisar o efeito do fator seletividade dos dados sobre o valor obtido a partir do método analítico de Theodoridis [36].
- Estender os métodos analíticos para abrangerem a análise de dados espaciais para espaços de alta dimensionalidade.
- Estudar as distribuições espaciais de dados sintéticas propostas por Ciferri de forma a avaliar se essas distribuições simulam dados reais de aplicações geográficas.
- Fazer um estudo para determinar qual a influência sobre os métodos analíticos do fator fixação das páginas de *buffer* nas junções espaciais.
- Desenvolver métodos analíticos mais genéricos para abranger outros tipos de predicados espaciais como direcionais e de distância.

Apêndice A

Apêndice

A.1 Representação Gráfica dos resultados obtidos

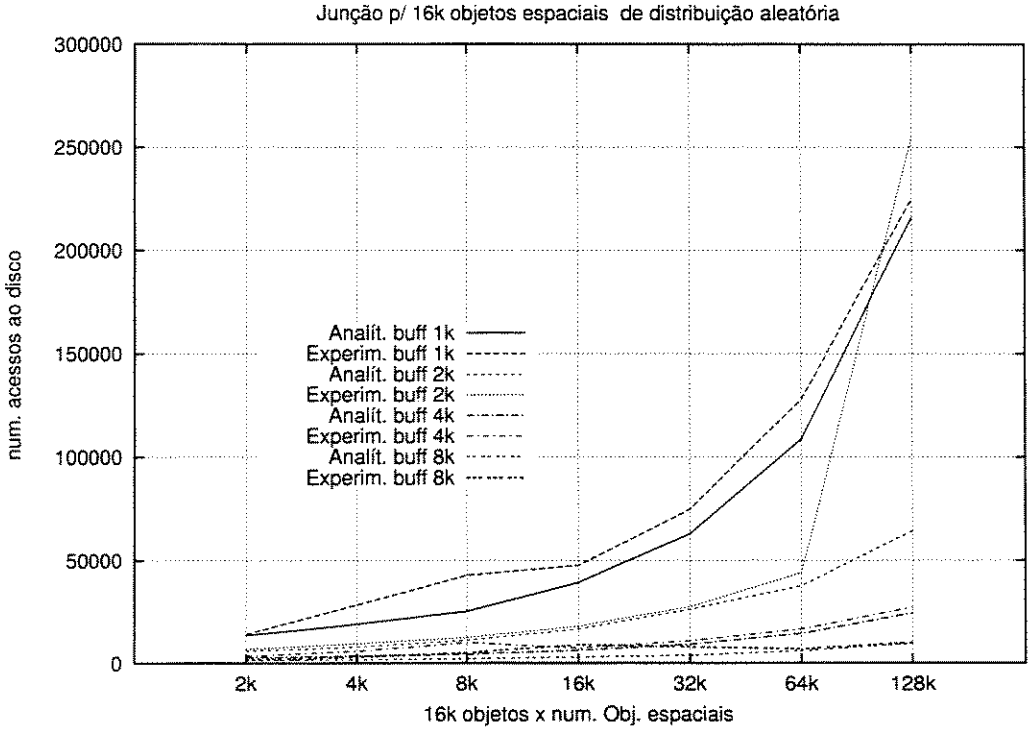


Figura A.1: Gráfico da junção entre o arquivo de 16k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

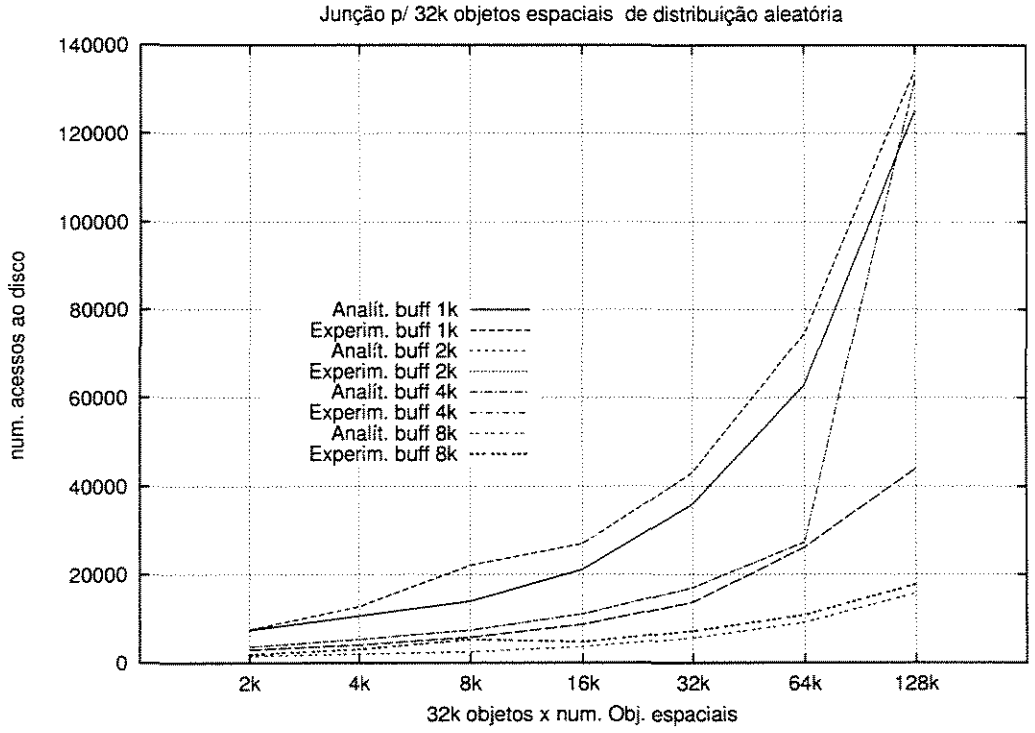


Figura A.2: Gráfico da junção entre o arquivo de 32k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

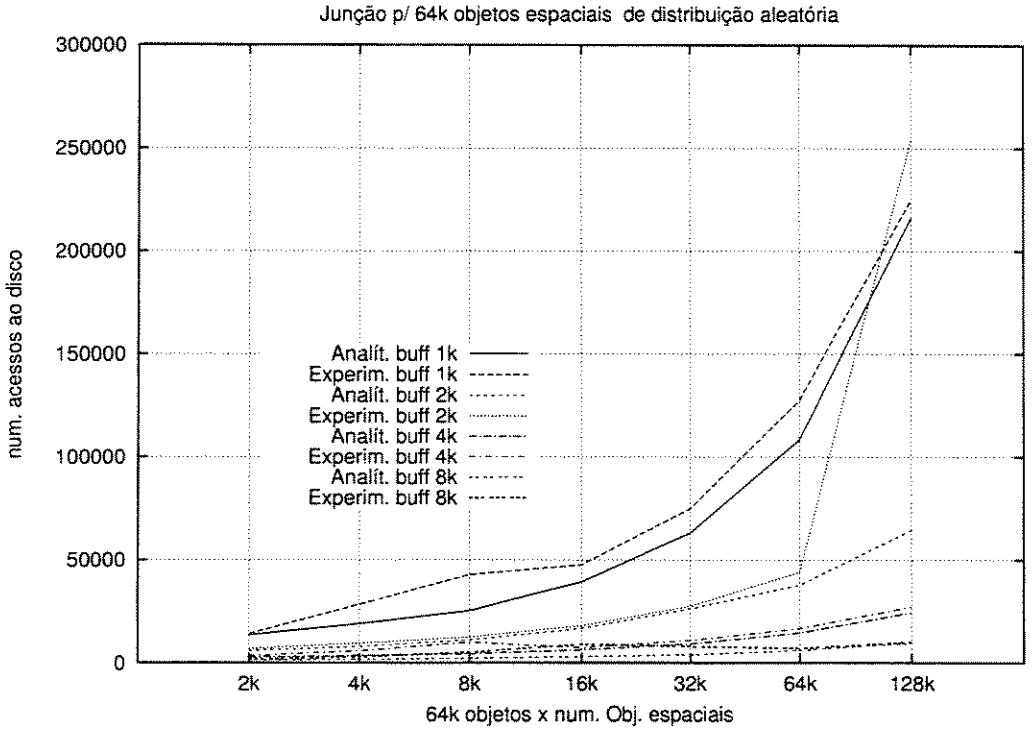


Figura A.3: Gráfico da Junção entre o arquivo de 64k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

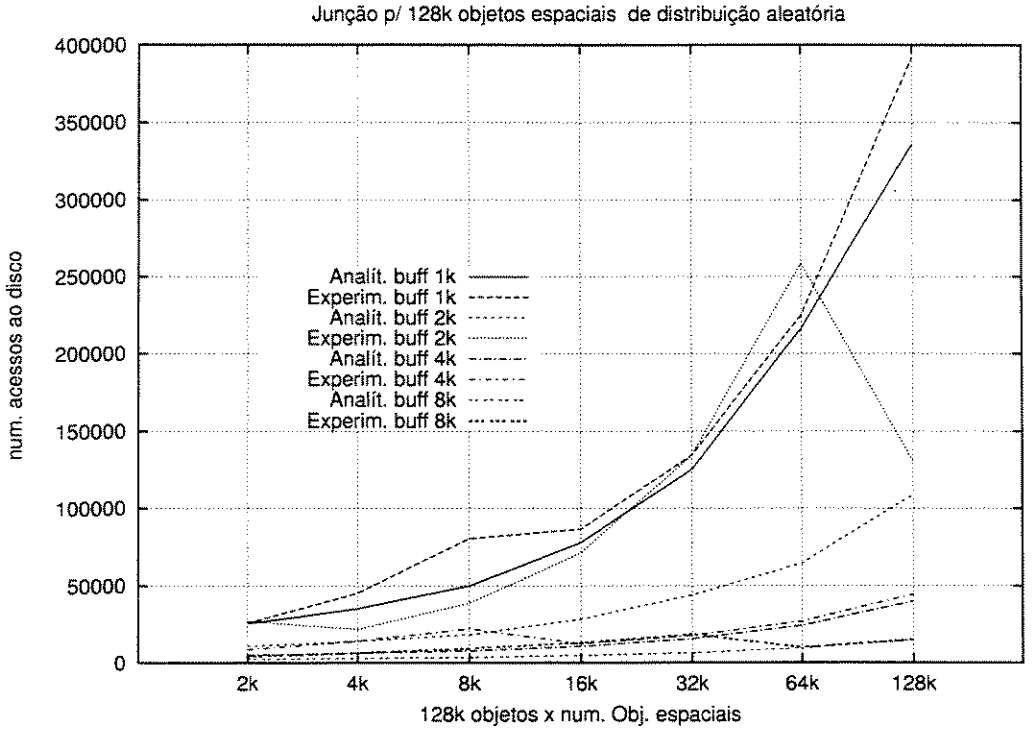


Figura A.4: Gráfico da junção entre o arquivo de 4k objetos e os arquivos de 2k, 4k, 8k, 16k, 32k e 128K objetos (Dados sintéticos simulando distribuição uniforme)

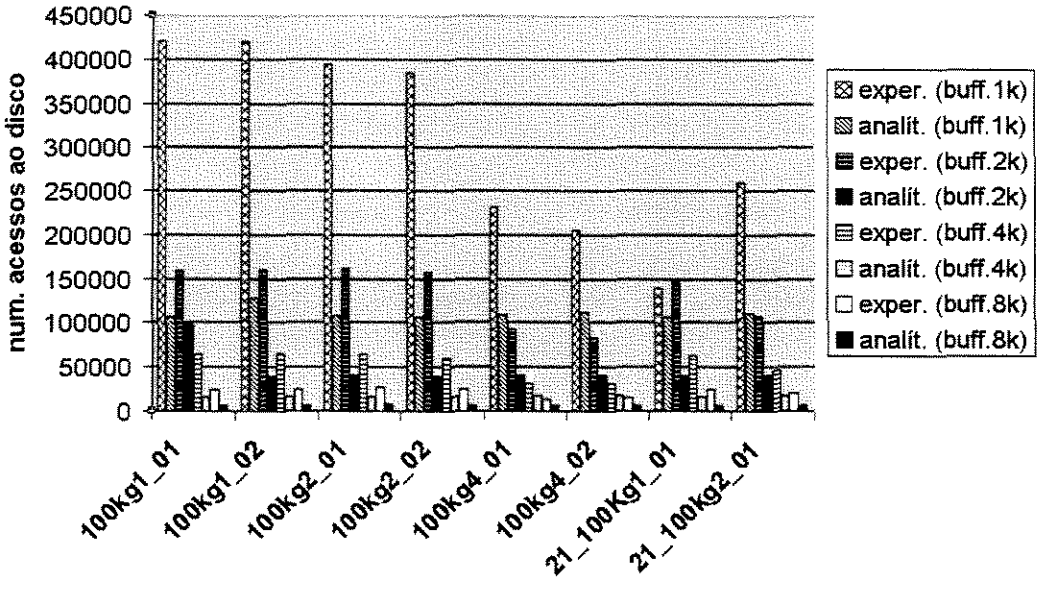


Figura A.5: Gráfico da junção entre o arquivo 100k_g1_01 e os arquivos 100k_g1_01, 100k_g1_02, 100k_g2_01, 100k_g2_02, 100k_g4_01, 100k_g4_02, 21_100k_g1_01 e 21_100k_g1_02 (Dados sintéticos de 100k objetos simulando distribuição não uniforme segundo o modelo de distribuição de dados de ciferri)

A.2 Visualização dos arquivos de dados utilizado nos testes

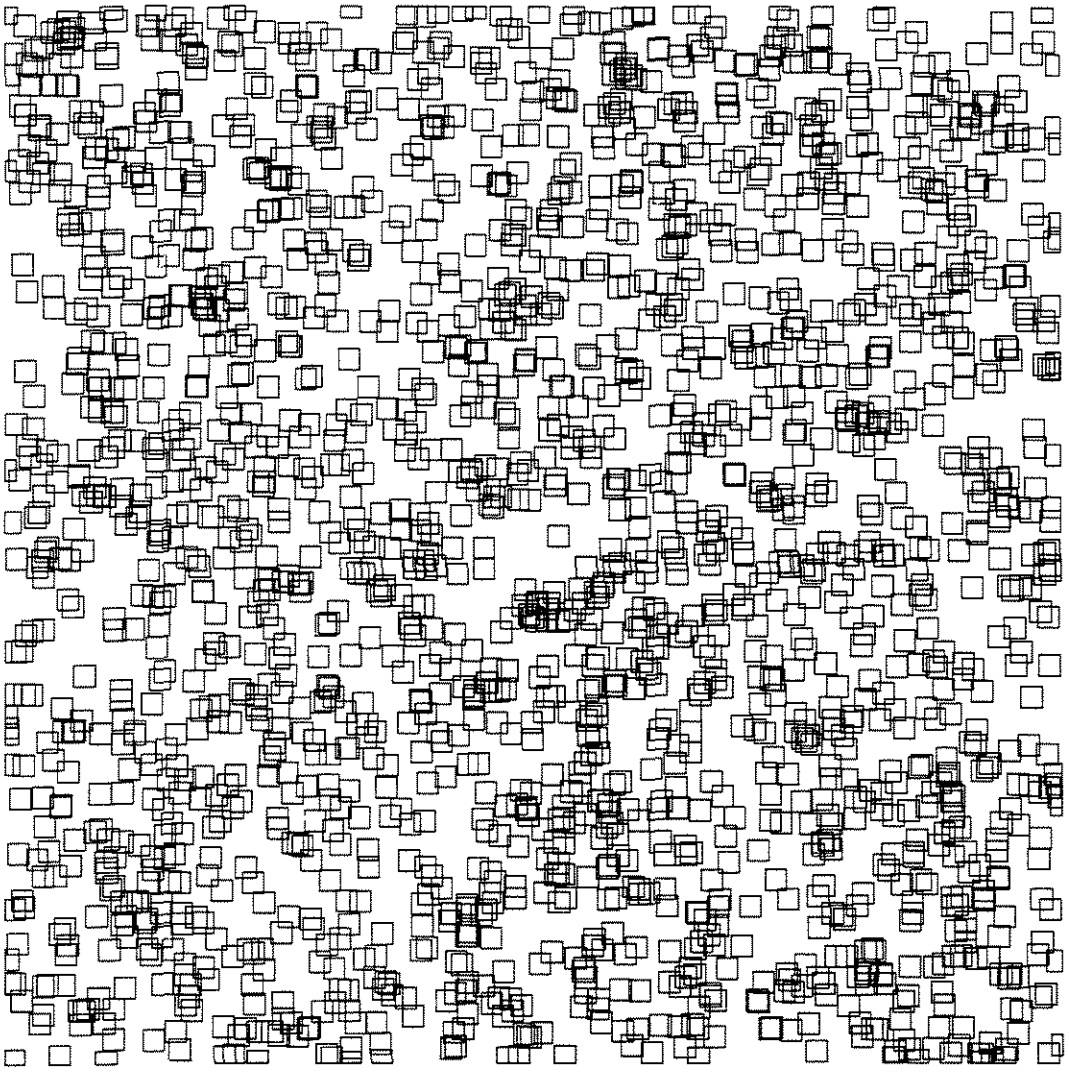


Figura A.6: Arquivo de dados randc2000.bin

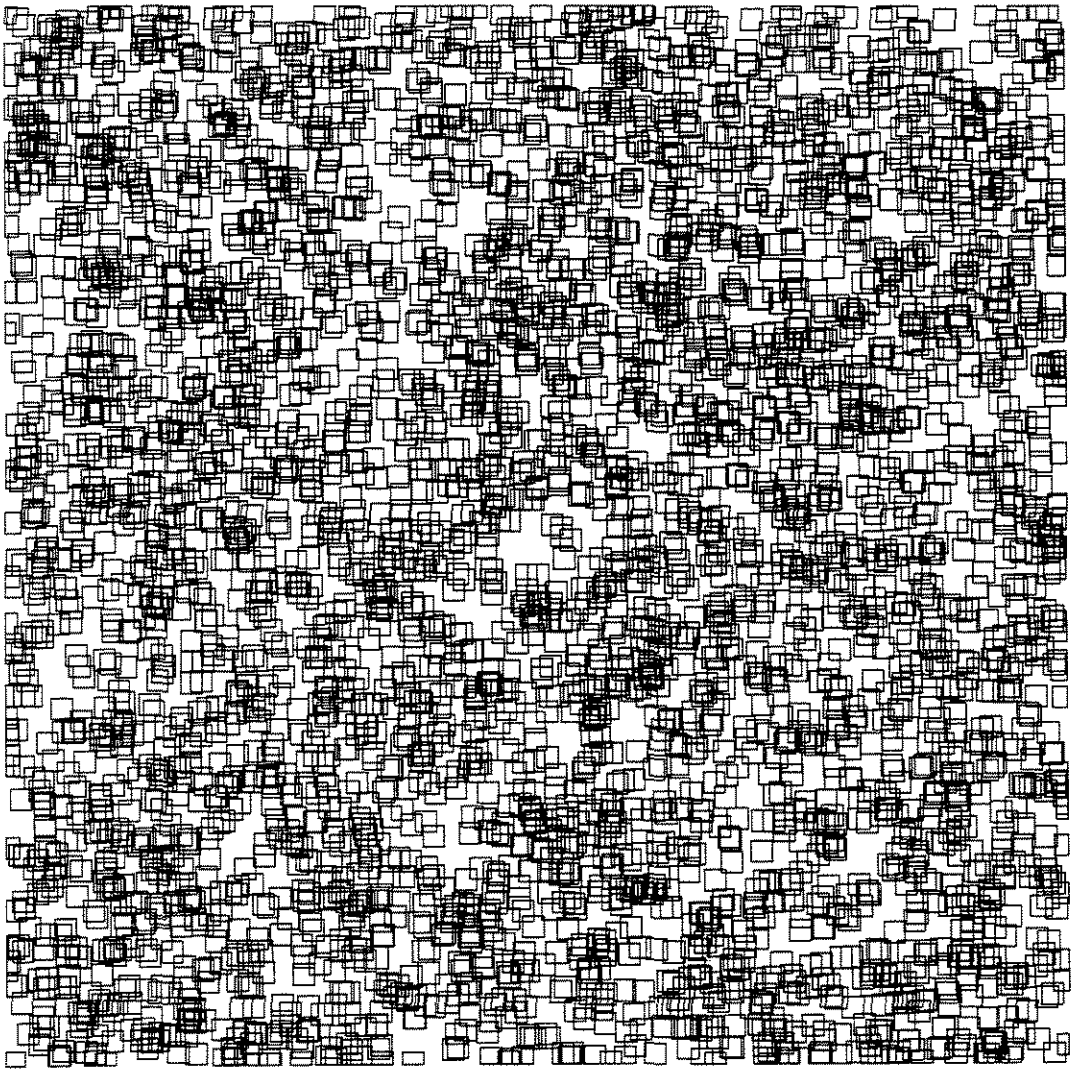


Figura A.7: Arquivo de dados randc4000.bin

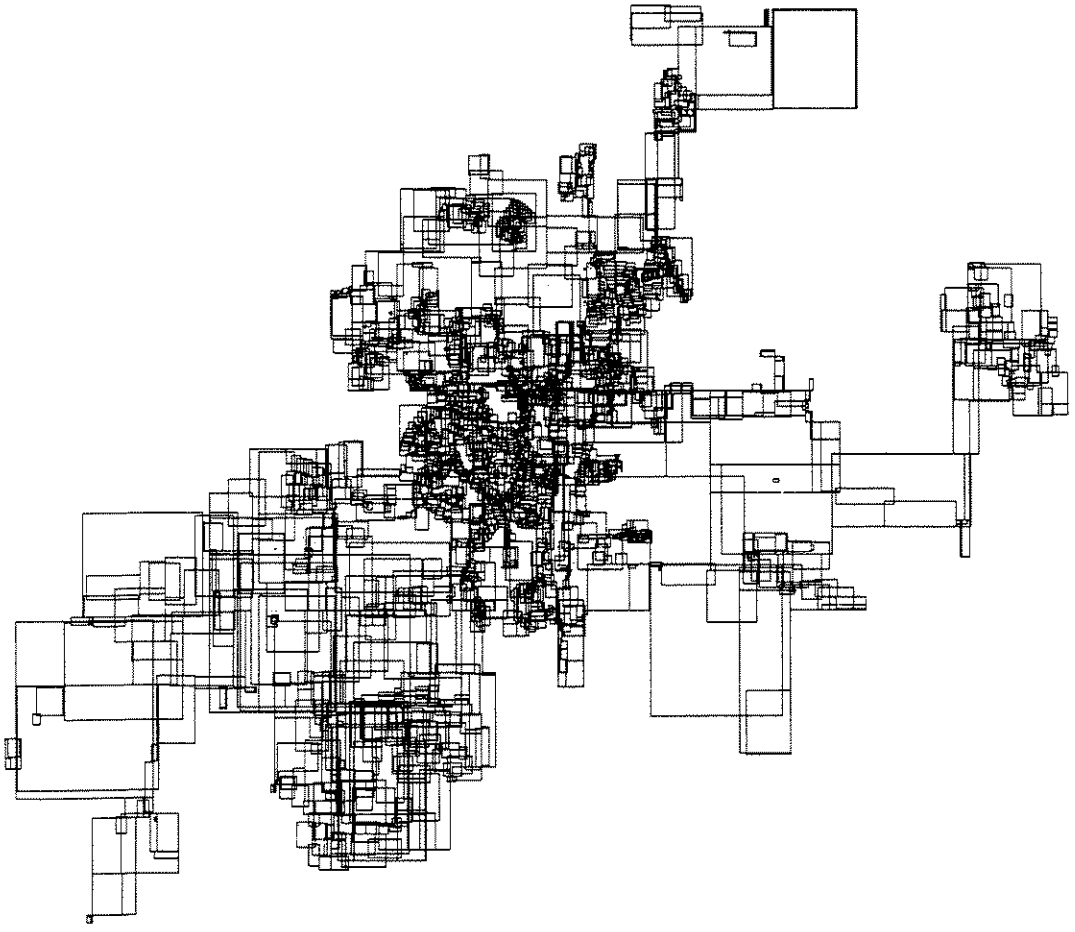


Figura A.8: Arquivo de dados reias (Quadras da cidade de Valinhos)

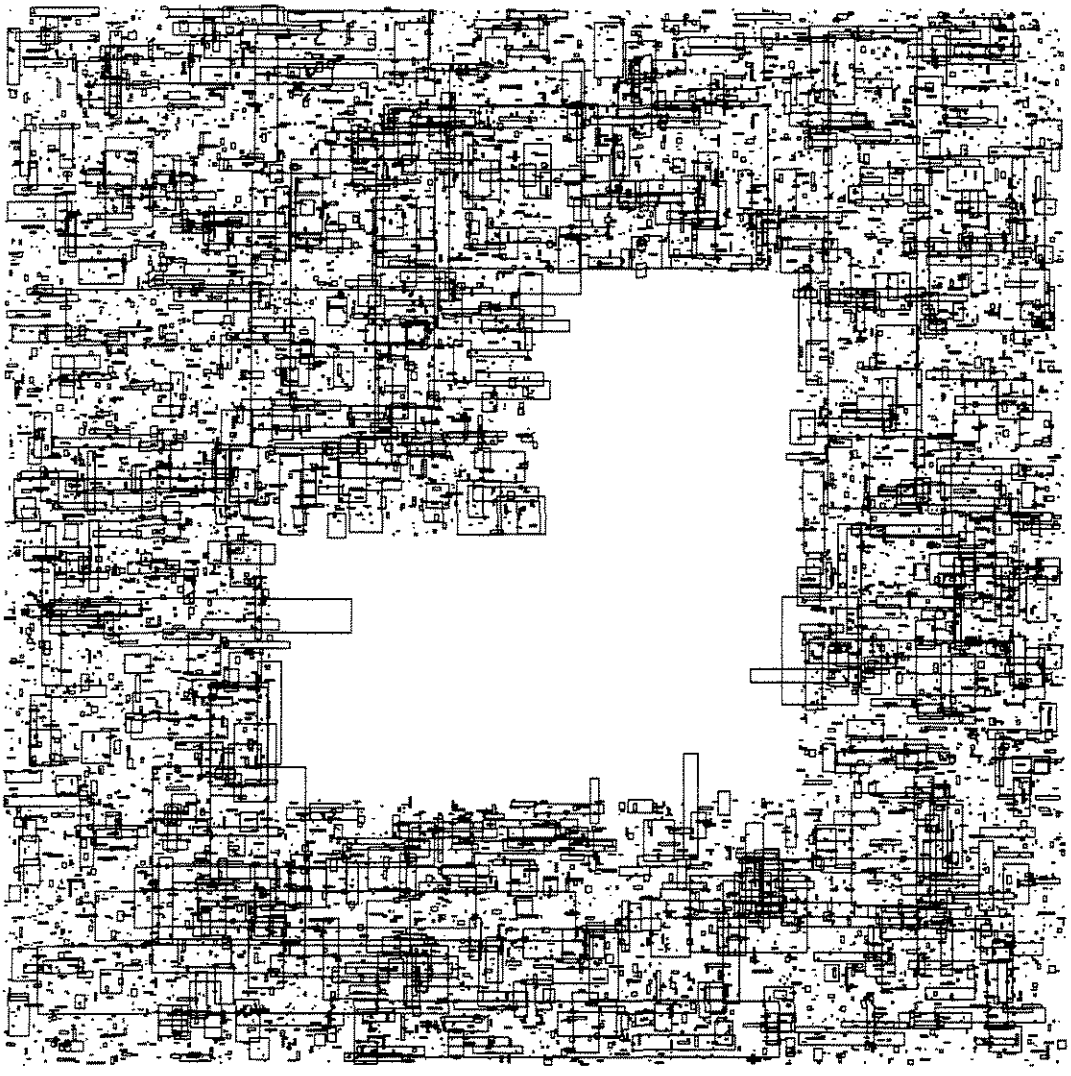


Figura A.9: Arquivo de dados D_SL20_COrg1_10k_retangulos_01 (10K_g1_01)

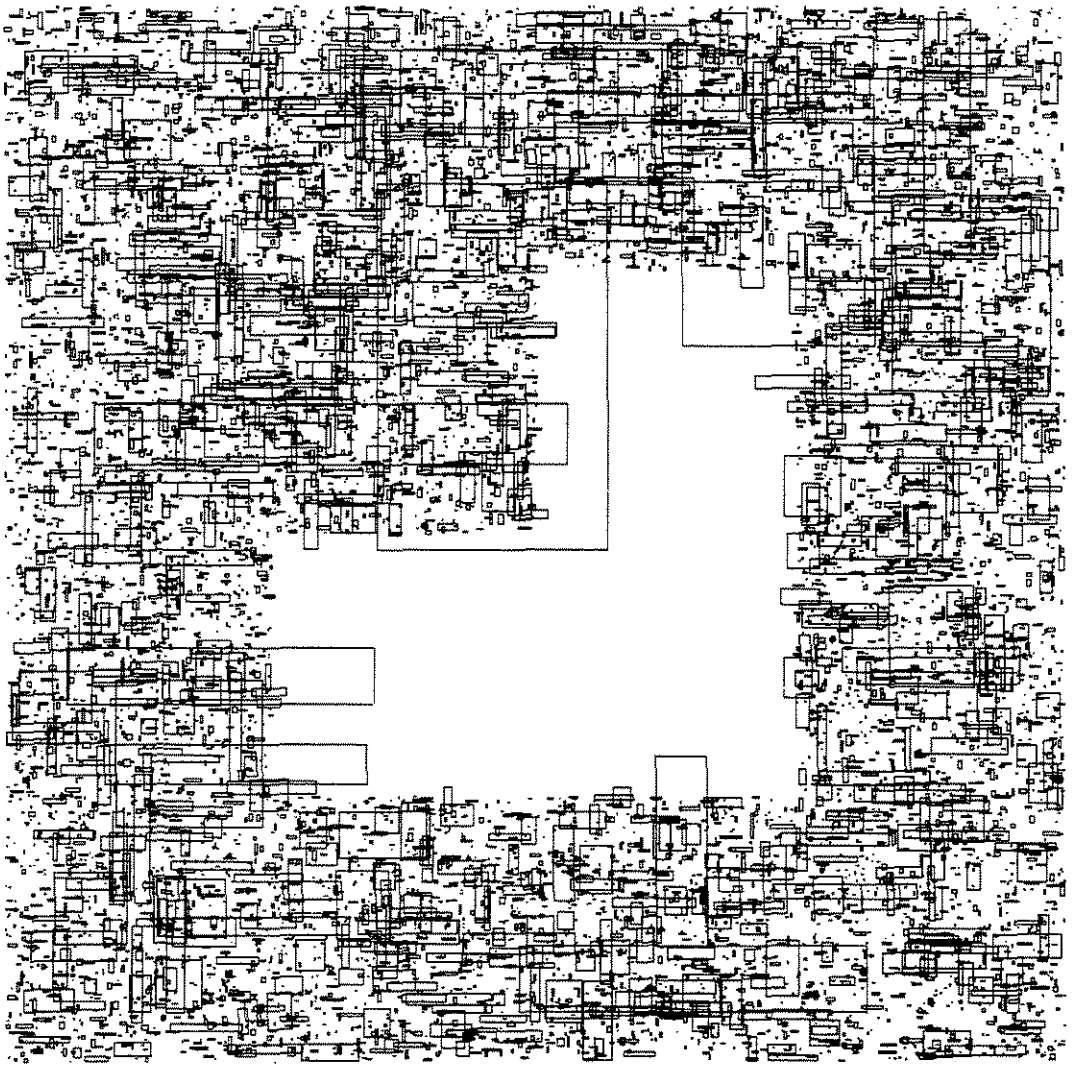


Figura A.10: Arquivo de dados D_SL20_COrg1_10k_retangulos_02 (10K_g1_02)

Figura A.11: Arquivo de dados D_SL20_COrg2_10k_retangulos_01 (10K_g2_01)

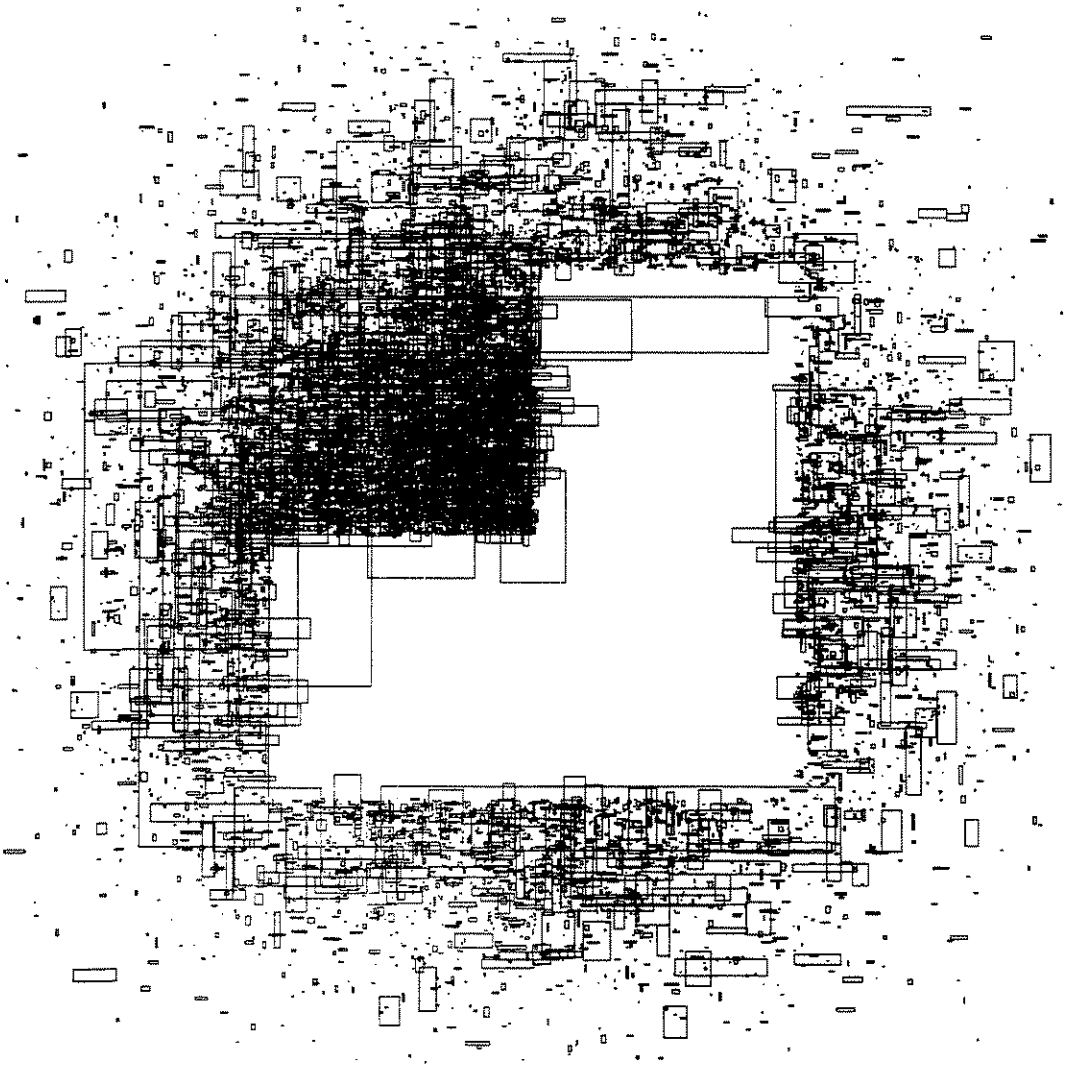


Figura A.12: Arquivo de dados D_SL20_COrg2_10k_retangulos_02 (10K_g2_02)

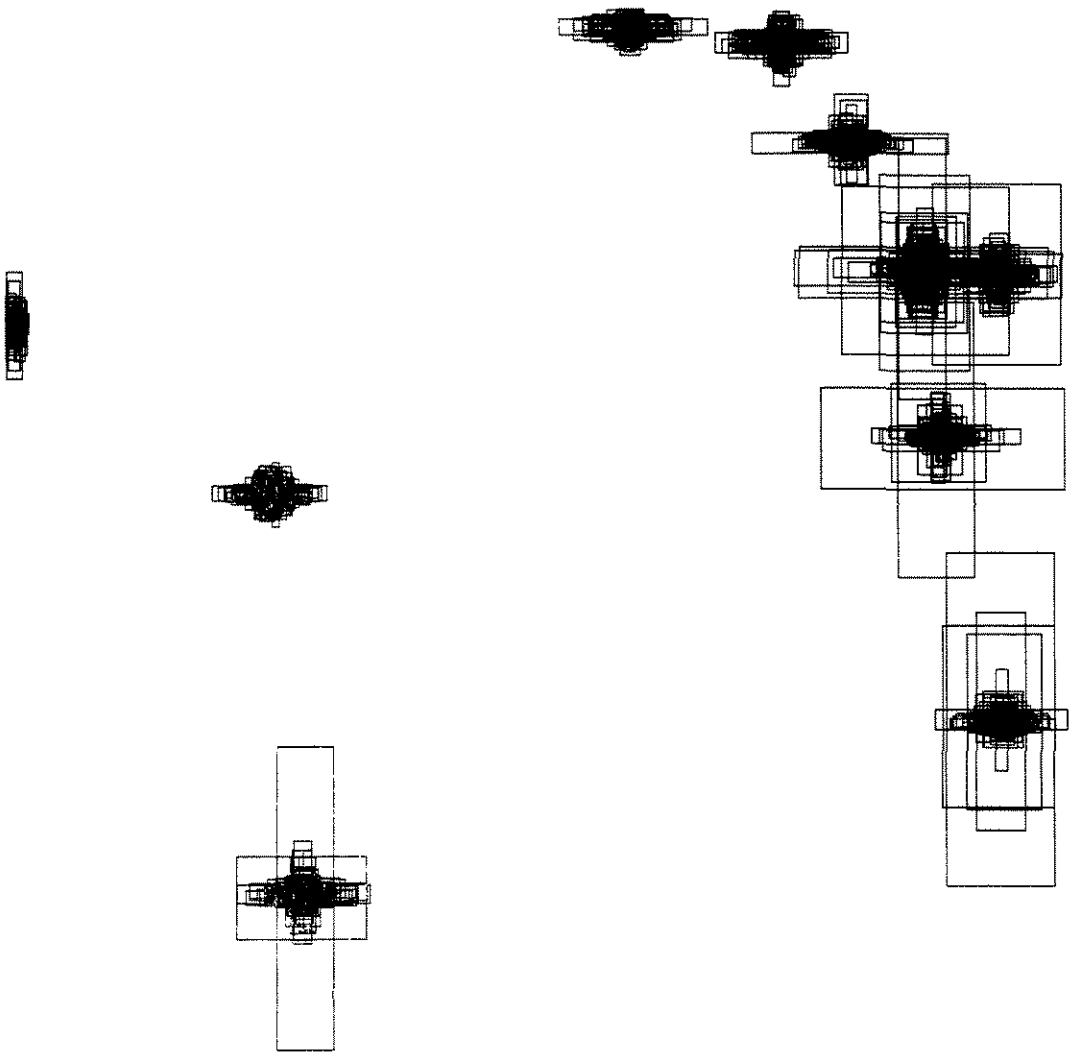


Figura A.13: Arquivo de dados D_SL20_COrg4_10k_retangulos_01 (10K_g4_01)

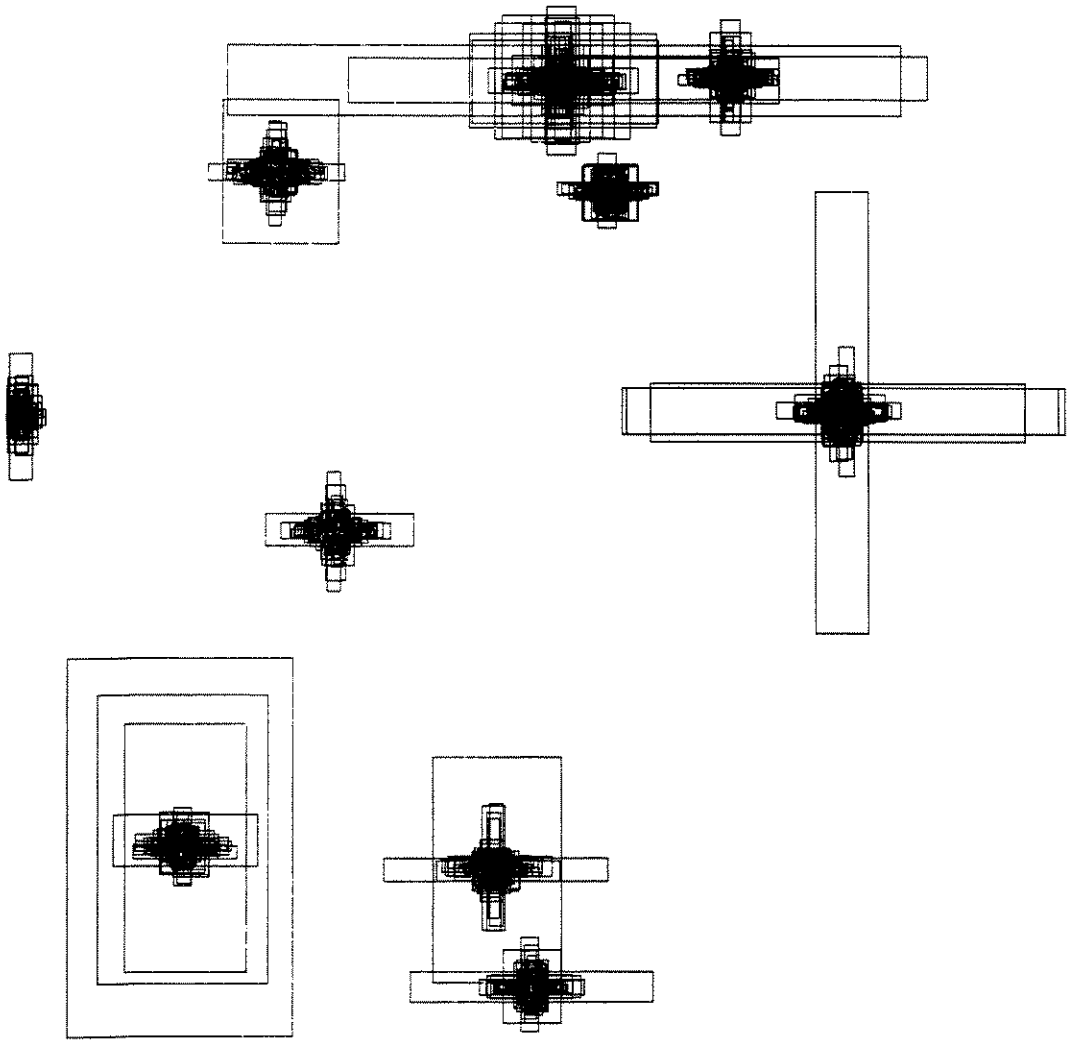


Figura A.14: Arquivo de dados D_SL20_COrg4_10k_retangulos_02 (10K_g4_02)

Referências Bibliográficas

[1] Ibm informix r-tree index user's guide. 10th International Conference on Scientific and Statistical Database Management, março de 1998.

[2] N. Beckmann, H. Kriegel, R. Schneider e B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. Em *Proceeding of the 1990 ACM SIGMOD International Conference on Management of Data*, Junho de 1990.

[3] Thomas Brinkhoff, Hans-Peter Kriegel, Ralf Schneider e Bernard Seeger. Multi-Step Processing of Spatial Joins. Em *In Proceeding of ACM SIGMOD Int Conf. on Manangement of Data*, 1994.

[4] Thomas Brinkhoff, Hans-Peter Kriegel e Bernhard Seeger. Efficient Processing of Spatial Joins Using R-trees. Em *Procedings of ACM SIGMOD Conf.*, 1993.

[5] A. P. Carneiro. Análise de desempenho de métodos de acesso espaciais baseada em um banco de dados real. Tese de Mestrado, Universidade Estadual de Campinas - UNICAMP, 1998.

[6] Ricardo Rodrigues Ciferri. *Análise da influência do fator distribuição espacial dos dados no desempenho de métodos de acesso multidimensionais.* Tese de Doutorado, Universidade Estadual de Maringá, 2002.

[7] R.R. Ciferri. Um benchmark voltado à análise de desempenho de sistemas de informações geográficas. Tese de Mestrado, UNICAMP, 1995.

[8] N.G. Colossi. Avaliação de desempenho de estruturas de acesso a dados hiperdimensionais. Tese de Mestrado, UNICAMP, 2000.

[9] F. S. Cox e G. C. Magalhães. Implementação e análise de métodos de acesso a dados espaciais. Em *VII Simpósio Brasileiro de Banco de Dados*, Porto Alegre, Maio de 1992.

[10] Christos Faloustsos e Ibrahim Kamel. Beyond Uniformity and Independece: Analysis of R-tree using the Concept of Fractal Dimension. Em *In Proceedings of the 13th ACM Symposium on Principles of Database Systems (PODS)*, 1994.

[11] C. Faloutsos, T. Sellis e N. Roussopoulos. Analysis of object oriented spatial access methods. Em *Prodeeding of ACM SIGMOD Conference on Manangement of Data*, 1987.

[12] Sandro Danilo Gatti. Fatores que afetam o Desempenho de Métodos de Junções Espaciais: um Estudo Baseado em Dados Reais. Tese de Mestrado, UNICAMP, Junho de 2000.

[13] Oliver Günther. Efficient computation of spatial joins. Em *Proceedings of the Ninth International Conference on Data Engineering, April 19-23, 1993, Vienna, Austria*, pp. 50–59. IEEE Computer Society, 1993.

[14] Antonin Guttan. R-Trees A Dinamic Index structure for spatial seaching. Em *Proc. SIGMOD ACM Conf.* ACM, 1984.

[15] Yun-Wu Huang e Ning Jing. A Cost Model for Estimating the Performance of Spatial Joins Using R-trees. Em *In Proceedings of 9th International Conference on Information and Knowledge Manangement (CIKM)*, 1997.

[16] Yun-Wu Huang e Ning Jing. Spatial Joins Using R-trees: Breadth-First Transversal with Global Optimizations. Em *Proceedings of the 23rd VLDB Conference Athens, Greece*, 1997.

[17] Yun-Wu Huang, Ning Jing e Elke A. Rundensteiner. Bfrj: Global optimization of spatial joins using r-trees. Relatório técnico, Dept. of Computer Science, Worcester Polytechnic Institute, jan de 1997.

[18] Chuck Murray Jeff Hebert. Oracle spatial user's guide and reference. Oracle Spatial User's Guide and Reference.

[19] I. Kamel e C. Faloutsos. Hilbert R-tree: an improved R-tree using fractals. Em *Proceedings of the 20th VLDB Conference*, pp. 500–509, Setembro de 1994.

[20] Ibrahim Kamel e Christos Faloutsos. On Packing R-tress. Em *In Proceedings of the 2nd International Conference on Information and Knoledge Manangement CIKM*, 1993.

[21] Scott T. Leutenegger e Mario A. Lopez. The Effect of Buffering on the Performance of R-trees. Em *In Proceedings of 14th ACM Symposium on Principles of Database Systems (PODS)*, 1998.

[22] G. C. Magalhães, A. V. Gigliotti, C. L. F. Santos, D. Teijiro e E. L. Argondizio. Especificação técnica da conversão de dados - Proposta da Telebrás - Projeto SAGRE. Em *GIS Brasil (Curitiba)*. 1994.

[23] G. C. Magalhães. Telecommunications outside plant managements throughout Brazil. Em *Proc. of XVII (Nashville 97)*.

[24] G. C. Magalhães. Projeto SAGRE. *Fator GIS*, 1993.

[25] G. C. Magalhães. The development of open systems for engineering applications. Em *Proc. of XVII Intl. Conference on AM/FM (Denver - Colorado)*, 1994.

[26] J.-M. Saglio O. Günther, P. Picouet. Benchmarking spatial joins À la carte. *International Journal of Geographical Information Science*, 1999.

[27] Bernard-Uwe Pagel e Hans-Werner Six. Are Window Queries Representative for Arbitrary Range Queries? Em *In Proceedings of 15th ACM Symposium on Principles of Database Systems (PODS)*, 1996.

[28] Bernard-Uwe Pagel, Hans-Werner Six, Heinrich Toben e Peter Widmayer. Towards an Analysis of Range Query Performance in Spatial Data Structures. Em *In Proceedings of the 12th ACM Symposium on Principles of Database Systems (PODS)*, 1993.

[29] Bernard-Uwe Pagel, Hans-Werner Six e Mario Winter. Window Query-Optimal Clustering of Spatial Objects. Em *In Proceedings of 14th ACM Symposium on Principals os database Systems (PODS)*, 1993.

[30] Dimitris Papadais, Nikos Mamoulis e Yannis Theodoridis. Processing and Optimization of Multiway Spatial Joins Using R-trees. Relatório técnico, HKUST-CS98-16, 1998.

[31] Ho-Hyun Park, Guang-Ho Cha e Chin-Wan Chung. Multi-way Spatial Joins using R-trees: Methodology and Performance Evaluation. Relatório técnico, KAIST Department of Computer Science, Jan de 1999.

[32] Jignesh M. Patel e David J DeWitt. Partition Based Spatial-Merge Join. Em *Proceedings of the 1996 ACM SIGMOD Int. Conf. on Manangement of Data*, junho de 1996.

[33] T. Sellis, N. Roussopoulos e C. Faloutsos. The R^+-tree: a dynamic index for multi-dimensional objects. Em *Proceedings of the 13th VLDB Conference*, pp. 507 – 518, Brighton – Inglaterra, Setembro de 1987.

[34] Yannis Theodoridis, Emmanuel Stefanakis e Timos Sellis. *An Efficient Cost Model for Spatial Joins Using R-trees.* Tese de Doutorado, National Technical University of Athens, 1997.

[35] Yannis Theodoridis, Emmanuel Stefanakis e Timos Sellis. Cost Models for Join Queries in Spatial Databases. Em *ICDE'98.* IEEE, Fev de 1998.

[36] Yannis Theodoridis, Emmanuel Stefanakis e Timos Sellis. Efficient Cost Models for Spatial Queries Using R-Trees. Em *IEEE Transactions on knowledge and data engineering*, volume 12, janeiro de 2000.

[37] Yannis Theodoris e Timos Sellis. A Model for the prediction of the R-tree Performance. Em *In proceeding of 15th ACM PODS Symposium*, 1996.

[38] Hein M. Veenhof, Peter M. G. Apers e Maurice A. W. Houtsma. Optimization of Spatial Joins Using Filters. Em *In Advances in Databases, 13th British National Conference on Databases, BNCOD*, 1995.